selenium testing

Overview

This page is the central entry point for information related to the Tiki Selenium testing project.

The goal of this project is to come up with simple infrastructure and procedures that will allow us to write automated acceptance tests for Tiki using Selenium WebDriver.

Below, you will find links to:

- A Tutorial for how to use Selenium to write tests for your own customer's sites, or for one of the *.tiki.org sites.
- Design Choices that were made in the process of building those testing procedures and infrastructures.

Also on this page, you will find the details of a User Centered Design exercise that was done to try and capture the needs of different people who might need to use the testing infrastructure.

The page also provides an implementation plan for how to build an increasingly powerful testing infrastructure. As of May 23, 2013, the goals of the first iteration of this plan have been achieved, although in a very different way than planned. The result is documented in the Tutorial

Tutorial: How to write and use automated tests for your own site

Click here to learn how to use the current procedures and infrastructure to write tests for your own customer's sites, or for one of the *.tiki.org sites.

Design Choices

Click here to learn about the rationale for the various choices we made in the process of building our testing procedures and infrastructures.

User Centered Analysis

Below is an analysis of the different type of people who will need this infrastructure, and what they will want to do with it.

Types of Users

Nelson the Tiki Consultant

Context:

- Have several highly customized instances of Tiki that they setup for customers.

Motivations:

- Avoid breaking his customer's sites when upgrading to trunk.
- Avoid making a configuration change that interferes with other configuration changes he made earlier.

Needs to do:

- Write tests on a customer's instance to capture that customer's key use cases.
- Setup a server that will run those custom tests once a day, and notify them when it fails one of them.
- After changes in the configuration, run those tests on a clone of the customer's current installation, to make sure that those config changes haven't broken any
Alain the Tiki Developer

Context

- Has certain parts of Tiki for which they are kind of "custodians".
- Creates and modifies code that might break someone else's code

Motivations

- Know when someone breaks "his" code.
- Know when he breaks "someone else's" code

Needs to:

- Write "generic" tests (i.e. not tied to a particular customer instance) to make sure that "their code" will keep working in the future.
  - Want some server somewhere to run those tests on trunk once a day, and notify them when it fails one of them.
- When modifying code in a part of Tiki, run a "short" (say 5 mins) test suite that covers the portions of the code that are most likely to be affected by the change.

Lindon, the Tiki Testing Power User

Context

- Knows Tiki very well, and may be site admin on several sites.
- Is not a programmer and may not have SSH access

Motivations

- Avoid breaking his sites, when upgrading to trunk
- Help the Tiki community, eventhough he is not a programmer

Needs to:

- Write tests on the sites he maintains, that captures the use cases he and his users care about.
- Have access to a server somewhere that can run those tests on trunk and notify him when one of them fails
  - May not be able to set it up himself.
- Write "generic" tests (i.e. not tied to a particular customer instance) on a test instance of Tiki.
  - May not be able to set one up himself.

Use Cases

Note: The use cases below describe an "ideal" system. Of course, we most probably will never reach that ideal, knowing what the ideal is gives us a clear direction to aim towards.

Further below, we will choose a small subset of the functionality described here, which can be implemented as a "Simplest Thing That Could Possibly Work" version of the ideal.
Record a test for a customer site

- Nelson is customising a Tiki site for a customer called "ConsultansRUs". He has just finished creating a complex workflow to help them with contract negotiation by tracking the state of a contract (draft, submission to client, signature etc...).
- He wants to write some tests that capture what the users will do with that tracker workflow.
- He clones his customer's DB into a test DB.

TODO: Implement an easy to use DB cloning tool

- He goes to tiki-test.pl, and clicks on Create snapshot. This opens a dialog that asks for the snapshot name. Nelson enters "ConsultansRUs" and clicks on Create.

TODO: Implement Frontend and backend for creating a snapshot from the current DB state

- From now on, Nelson will be able to create tests that start from that snapshot state.
- Nelson is now ready to record the new test.
- He goes to tiki-test.php, and clicks on Record a test.
- This starts a wizard that will guide him through the steps of recording the test.
  - Also, from then on, every Tiki page will have a header at the top with fields that are useful for testing purposes (ex: for entering comments about the test).
  - Step 1 of the wizard says: "Start your Selenium IDE recorder" (with a link to some help about Selenium IDE).

TODO: Implement a testing header which will only be visible when the system is in test recording mode.

  - Nelson starts Selenium IDE recorder.
  - Step 2 says: "Select a DB snapshot to start from".
    - The list contains LOTS of possible snapshots, but fortunately, that list comes with a search form. He types "ConsultantsRUs" and finds the snapshot he just created earlier. Nelson clicks on it and the system restores the DB to that start state

TODO: Implement backend and frontend functionality capable of quickly restoring a snapshot

  - Step 3 says: "Carry out the actions that you want to record, and click on "End of test" button when done (that button appears in the testing header).
  - Nelson then carries out the various actions involved in the test scenario, adding some assertions at specific points. Also, at each new screens, he enters some comments in the Comments field of the testing header, describing the intent of the actions that will happen in that screen (in other words, what is he trying to test with those actions). This gets recorded by Selenium as a typing event. Developers will then be able to see those comments in the Selenium script. It's a hack, but it's simple and I think it will do the job).

TODO: Add a comments box in the testing header.

  - When he is done, he hits the End of test button in the testing header.
  - Step 4 is displayed. It says: Stop Selenium IDE, and save the recorded test to your hard drive (with a link to some help on how to do this with Selenium), and upload that file.
  - Nelson saves the test to disk (using the Selenium IDE). He clicks on the Upload test button, and browses to the test file.
  - This opens up a dialog that displays a list of existing tags for the test. None of the existing tags is relevant, so Nelson creates a new one called "Contract negociation workflow", and tags the test with that. That way, if ever he wants to run only those tests that relate to the contract negociation workflow, he can do so by running tests with that tag.

TODO: Implement a dialog for assigning tags to tests, and creating new ones.

  - For Test sharing, Nelson selects "Private", which really means "Put that test in the "custom" directory which is not shared on SVN".
  - Nelson then clicks on the Upload button, and the test ends up somewhere in the custom directory of the testing clone. From there, Nelson can commit his custom stuff to a private GitHub repository or
something of the sort. Next time he commits that custom directory to his VCS system, the test will be committed.

TODO: Implement a feature for uploading the test to custom, and making sure that it's stored in a way that will make it retrievable using the selected tags.

does that. He now has an HTML file that describes the test in the Selenium language.

Record a generic, non-customer specific test (Alain variant)

- Alain is working on adding a new type of Captcha in Tiki, and he wants to create some tests that capture what can be done with this new Captcha type.
- On his local dev instance of tiki, he goes to tiki-test.pl, and gets to the testing wizard.
- Step 1 of the wizard says: "Start your Selenium IDE recorder" (with a link to some help about Selenium IDE).
  - Alain starts Selenium IDE recorder.
- Step 2 says: "Select a DB snapshot to start from".
  - Alain browses and searches through the list of potential starting points (each of which has a short description), and he finds one that seems like it should have Captchas activated (ex: "Self registration site"). He chooses that one.
- At that point, the site is configured in a way that is appropriate for recording a test for Captchas.
- Step 3 says: "Carry out the actions that you want to record, and click on "End of test" button when done (that button appears in the testing header).
- Alain then carries out the various actions involved in the test scenario, adding some assertions and comments at specific points.
- When he is done, he hits the "End of test" button.
- Step 4 is displayed. It says: Stop Selenium IDE, and save the recorded test to your hard drive (with a link to some help on how to do this with Selenium).
- Alain uses Selenium IDE to save the test to his HD.
- He then uploads the test file (using the tiki-testing.php wizard) and tags it with a new tag that he creates, called "Captcha". He also tags it with other relevant tags like "Registration", "Spam protection" and "Security".
- For Test sharing, he selects "public", which means that it will be put in a directory that is under SVN control.
- Alain clicks the final Upload button, and the next time he commits to trunk, this test, along with the code for the new Captcha type will be committed.

Record a generic, non-customer specific test (Lindon variant)

- Lindon wants to help the Tiki community by writing test for the Blogging feature.
- This scenario plays out pretty much the same way as the Alain variant, except that Lindon does not have SSH access, and therefore it may not be possible for him to commit his new test and the new snapshot to trunk.
- Maybe Lindon needs some kind of SVN web interface to his own installation.

TODO: Investigate whether it's possible to provide Lindon with a SVN web interface to his own tiki installation (commit from web???)

Create a customer specific snapshot from a customer instance

- Nelson has a highly customised Tiki instance for a given customer. He wants to create a snapshot that corresponds to the current state of that Tiki instance.
- That scenario is described at the beginning of the use case Record a test for a customer site above (I think this pretty much covers all possibilities).
Create a generic snapshot

- Alain (or Lindon for that matter) want to create a snapshot that can be used to record a series of generic tests.
- Alain goes to tiki-test.php, and browses the list of existing snapshots. He finds one that might be a good starting point for creating the new snapshot. Say it's called Blog testing. If none of the existing snapshots seem like good starting points, he can always use the default one, which is called barebone install.
- Starting from that state, Alain further configures the Tiki install, possibly adding some test data (blog posts, wiki pages, etc...) that will be needed for the tests.
- Once he is satisfied that the DB is in a state appropriate for the tests, he clicks on Save snapshot, and provides a new name for that snapshot.

Modify an existing snapshot

- Alain created a snapshot for testing blogs, called Blog testing. This snapshot does fine for the first few tests, but eventually, he finds he needs the snapshot to be slightly different (maybe it needs to have a few more blog posts that are handy for testing).
- So he needs to modify the snapshot.
- He goes to tiki-test.php, and chooses the Blog testing snapshot.
- He configures the snapshot further, then clicks on Save button, without changing the snapshot name (so it stays at Blog testing). The system asks Alain if he wants to overwrite the existing Blog testing snapshot and Alain answers yes.

Run a relevant subset of all tests (Nelson and Alain variants)

- Because the tests take 8h to run, it will only be possible to run all of them once a day or so.
- But when they are making changes, Nelson, Alain and Lindon all need to have faster feedback on whether or not they broke something. So they need a simple way to select a relevant subset of the tests that cover most of the places that are likely to be affected by their changes, and yet will run in a reasonably short time.
- The process should look approximately the same for Nelson and Alain (but it will be different for Lindon... see further on this page).
- Open a terminal window and go to the Tiki installation's testing directory.
- Type something like:

  ```bash
  run_tests.php --tags="blog|articles" --sharing="private|public"
  ```

  - This will run all the tests that are tagged with either "blog" or "articles" and are either private or public, and provide the usual phpUnit report of what tests failed and why.
  - The run_tests.php command will also allow Nelson and Alain to choose a subset of tests to run, by telling them how many tests will be run

  ```bash
  run_tests.php --count_only --tags="blog|articles" --sharing="private|public"
  ```

  - Rather than run the selected tests, the above command will just tell Nelson or Alain how many tests meet the selection criteria.

TODO: Implement a run_tests.php script to run the Selenium tests.

One requirement of the run_tests.php command is that it must be able to run on an instance of Tiki that differs from the instance where the tests were recorded.
TODO: run_tests.php must be able to run tests on a Tiki instance that has a different URL from the Tiki instance where the tests were recorded.

Run a relevant subset of all tests (Lindon variants)
- Lindon has the same needs as Nelson and Alain, to select and run a relevant subset of the tests that will run in a reasonable amount of time.
- The problem is that Lindon does not have SSH access, so he can't use the run_tests.php command line.
- He needs to be able to run the tests through a web interface. But it's not clear how that can be done exactly.

TODO: Figure out how to run the tests of an installation, through the web interface of that installation. Or maybe we should use the web interface of another installation, but then how do you ensure that the tests are aligned with the code and snapshots?

Setup a daily run of all of a customer's tests
- Nelson has cloned his customer's site and created a bunch of custom tests that reside in a GitHub repository.
- He wants to setup a server that will run all those tests on a daily basis, and notify him of any failure.
- He wants to use SauceLabs.com for that.
- What are the steps?

TODO: Implement code for translating the Selenium IDE tests into php scripts. That's because SauceLabs.com does not support tests that are in the Selenium IDE native format. Or, if there is no PHP library for doing this conversion, provide instructions to the user for now to convert the Selenium file to PHP before uploading it.

TODO: Implement a script that can be run from a Tiki instance, which will

a) Upgrade the instance to trunk.
b) Upload the upgraded tests to SauceLabs.com.
c) Launch a daily SauceLabs.com run of the test on the instance.

This could be implemented as an option of run_tests.pl script. For example:

run_tests.php --action=upgrade_and_test --drive_from_saucelabs=<CREDENTIALS>

Where <CREDENTIALS> includes whatever credentials are needed to login to the saucelabs account from which the test are to be driven.

Setup a daily run of all the generic Tiki tests
- Nelson doesn't just care about Tiki running on his customer site. He also cares about the health of Tiki for future customers, and for the Tiki community at large.
- So he wants to setup a community server that will do a daily run of all the generic Tiki tests (as opposed to the customer-specific tests).
- He wants to use SauceLabs.com for that.

TODO: Figure out how to use SauceLabs.com to run Tiki generic tests, and document that process.

Upgrade all snapshots after a Tiki upgrade
- Alain has a localdev Tiki instance that he uses to run tests on. This instance has many snapshots some of which he created).
- Now Alain upgrades his Tiki instance, at which point, all of his snapshots are using the wrong DB schema, and need to be upgraded.
- Alain goes to tiki-test.php, and hits the Upgrade all snapshots button.
TODO: Implement a feature for upgrading all the snapshots.
- Note: Eventually, it would be nice if the snapshots were upgraded automatically the first time we try to load them after the Tiki upgrade.

TODO: Implement a mechanism whereby Tiki can tell whether a snapshot is up to date or not, and reload it as needed.

Modify an existing test
- For Nelson and Alain, probably would be good enough to be able to open the test in a regular editor, and reload it in the selenium IDE and modify it from there.
- For Lindon, we need a web based UI to allowing him to browse the test suite to retrieve the content of an existing test, to put it in the Selenium IDE.

TODO: Implement a web based UI for navigating the list of existing tests, and retrieve the source of a given test.

Miscellaneous issues
- Need a way to run tests that were recorded on one server, to run on a different server. In other words, remove the hard-coded base URL used in the Selenium tests, and replace it with a base URL provided by the person running the test.
- When thinking about the UI for creating and select DB snapshots, should look at the mac automator.
- The testing header (the place where you can enter comments for the test, etc...) may interfere with the tests. In other words, there may be tests that only fail when the header is present, or tests that only work when the header is present. If that happens too often, we may have to get rid of the header altogether.

Implementation plan
Below is a plan for building an increasingly flexible testing infrastructure for Tiki.

Note that as of May 23, 2013, Iteration 1 has been completed. It is documented in this tutorial

Iteration 1: Simplest Thing That Could Possibly Work for writing and running Customer specific tests

At the end of this iteration, Nelson will be able to create and run tests for a customer site and run them on SauceLabs.

The main limitations will be:
- Tests will have to be written in a "timeless" fashion, meaning that their expectations don't depend on whether or not other tests have been run previously.
  - In other words, it won't be possible to create and restore DB snapshots.
- The tiki instance where tests are run, will have to be the same tiki instance on which tests will be recorded
- It will not be possible to upload and tag recorded tests using the web interface.
  - In particular, this means test will have to be recorded on a site that is either a local dev machine, or is a machine to which he has FTP access.

TODO: Implement a run_tests.php script to run the Selenium tests.

Time estimate: Low: 1 day, High: 2 days, Actual: N/A
TODO: Implement code for translating the Selenium IDE tests into php scripts. That's because SauceLabs.com does not support tests that are in the Selenium IDE native format. Or, if there is no PHP library for doing this conversion, provide instructions to the user for now to convert the Selenium file to PHP before uploading it.

Time estimate: Low: 1 day, High: 2 days, Actual: N/A

TODO: Figure out how to use SauceLabs.com to run customer-specific tests, and document that process.

Time estimate: Low: 1 day, High: 2 days, Actual: N/A

TODO: Implement a script that can be run from a Tiki instance, which will
a) Upgrade the instance to trunk.
b) Upload the upgraded tests to SauceLabs.com.
c) Launch a daily SauceLabs.com run of the test on the instance.

This could be implemented as an option of run_tests.pl script. For example:

```
run_tests.php --action=upgrade_and_test --drive_from_saucelabs=<CREDENTIALS>
```

Where `<CREDENTIALS>` includes whatever credentials are needed to login to the saucelabs account from which the test are to be driven.

Time estimate: Low: 2 days, High: 4 days, Actual: N/A

Iteration Total: Low: 5 days, High: 10 days, Actual: N/A

Iteration 2: Generic, Community-based tests

At the end of this iteration, it will be possible for community members to write generic tests which are not customer specific, and contribute them to a community maintained test suite.

In particular, it will be possible to:

- Create tests on one tiki instance, that can be played back on any instance
- Create stable starting points for the tiki db, which can be used by various tests

The main limitations will be:

- It will not be possible to upload and tag recorded tests using the web interface.
  - In particular, this means test will have to be recorded on a site that is either a local dev machine, or is a machine to which he has FTP access.

TODO: run_tests.php must be able to run tests on a Tiki instance that has a different URL from the Tiki instance where the tests were recorded.

Estimate: Low: 1, High: 2, Actual: N/A.

TODO: Implement Frontend and backend for creating a snapshot from the current DB state

Time estimate: Low: 1 day, High: 2 days, Actual: N/A

TODO: Implement backend and frontend functionality capable of quickly restoring a snapshot

Time estimate: Low: 1 day, High: 5 days, Actual: N/A
TODO: Implement a feature for upgrading all the snapshots.

Time estimate: Low: 1 day, High: 2 days, Actual: N/A

Iteration total: Low: 4 days, 11 days, Actual: N/A

Iteration 3: Make the process and the resulting tests easier to use

At the end of this iteration, it will be possible to:

- Upload and tag new tests through the web interface
- Enter comments at each page of a recorded test

TODO: Implement a testing header which will only be visible when the system is in test recording mode.

TODO: Add a comments box in the testing header.

Time estimate: Low: 0.5 day, High: 1 day, Actual: N/A

TODO: Implement a dialog for assigning tags to tests, and creating new ones.

Time estimate: Low: 0.5 day, High: 1 days, Actual: N/A

TODO: Implement a feature for uploading the test to custom, and making sure that it's stored in a way that will make it retrievable using the selected tags.

Time estimate: Low: 1 days, High: 2 days, Actual: N/A

Iteration total: Low: 2 days, 4 days, Actual: N/A

Iteration 4: Extra goodies for Lindon

By Iteration 3, Nelson and Alain will probably have everything they need. But that won't be the case for Lindon, because:

- He has no easy way to commit tests he created.
- He has no easy way to select and execute tests (he can only run the one test that he is currently recording in Selenium IDE).

In Iteration 4, we will implement features that will allow Lindon to fully take advantage of the testing infrastructure.

**NOTE: These features are probably better implemented by Amette or someone with good sysadmin experience. We should consult him for time estimates.**

TODO: Implement an easy to use DB cloning tool. In particular, it should be easy to create a clone on show.tiki.org or test.tiki.org, for carrying out tests on.

Estimate: Low: 3, High: 6, Actual: N/A

TODO: Investigate whether it's possible to provide Lindon with a SVN web interface to his own tiki installation (ex: WebSVN?)

Estimate: Low: 1, High: 2, Actual: N/A
TODO: Figure out how to run the tests of an installation, through the web interface of that installation. Or maybe we should use the web interface of another installation, but then how do you ensure that the tests are aligned with the code and snapshots? Or maybe you use a community SauceLabs.com account?

Estimate: Low: 1, High: 2, Actual: N/A

TODO: Implement a web based UI for navigating the list of existing tests, and retrieve the source of a given test.

Estimate: Low: 1 day, High: 2 days

Iteration estimate: Low: 6 days, High: 12 days, Actual: N/A

Running total: Low: 20 days, High: 41 days, Actual: N/A

Nelson's notes

In the longer run the goal is to have our own Selenium testing infrastructure (which include the dashboard), but for now we have decided to use SauceLabs which provide Selenium infrastructure where we can set up testing for the tiki.org sites.

Pascal will contact SauceLabs to get it setup.

Also, we will need to do:

1) Document how to record tests using Selenium IDE (a Firefox plugin) to record tests. (Alain assigned)

2) Document where to share tests. Wiki pages will be fine for now but the idea is to have them with SVN but see next point (Alain)

3) Document how to get them into a "test suite" which will be run by the SauceLabs testing infrastructure (Alain)

4) Start creating tests for the tiki.org sites, especially doc or dev. (Let's get everybody together for this).

5) Longer term roadmap to have tests committed together with features in SVN (but since this would require getting instances to a certain configuration and with test content to run the tests from, it is dependent on the dev:Configuration Management and Systems Orchestration project to be done first.

6) Once the above is setup, Amette will figure our a way to automate SVN bisect to identify track what commit is behind which error.

7) Perhaps this can automatically be run on another machine on any error and the results inserted into a tracker on quality.tiki.org. Nelson will have coordinate with LPH how to insert it automatically into the specified tracker, probably by calling the AJAX tiki server,