Applied Twill

Abstract

Case study on using twill at Playlist.com.

Write a simple twill script to smoke test and monitor your web site. You need to install python and twill. You use a few of the simple but powerful twill commands in your script. Save time by not creating a test harness and instead use the operating system task scheduler to run the twill script every five minutes.

Background

Playlist.com is an internet based music discovery site. Users search for music, create and save playlists. Users can create accounts, upload photos, friend each other, chat, and interact with as in any online social network. Playlist made money through advertising and revenue share on ring-tone purchases.

When I first started at Playlist all searchable music had been linked to by users; Playlist hosted no music content, although it did cache links to the music files. At the time of this writing, Playlist is working with the major labels to host licensed content, ending its practice of linking to music through the internet. I joined in August, 2008, Playlist had about fifteen developers, and all testing was done by development.

The Problem

When I arrived at Playlist.com there was no release management and no qa. Developers frequently pushed their code to the production website. Frequently could mean several times a day or several times a week. Compounding this problem was that a developer who had pushed code to production might not inform all interested parties about what he had done. Examples of interested parties are other developers, qa, and technical support. As a result, the web site was often broken or down, but no one in engineering knew about it.

In addition, the IT/colo team was not properly staffed, and they were offsite (in southern California, while development was in Palo Alto). Even though munin and nagios were installed to monitor the system, often the first time engineering learned about a problem on production was through technical support, who had heard about the problem from an end user.

A partial solution

While a number of changes were clearly needed to the engineering process, I wanted to some basic automation to keep an eye on the production web site. The requirements were:

1. The automation had to be easy to script
2. The automation had to run quickly, e.g. traverse a number of pages within the web site in seconds or minutes

3. The automation had to be immune to frequent web site changes, not requiring modification every time a change was made to the site.

Since I was the only qa engineer, and there was an existing backlog of testing and an unrealistic dev to qa ratio, I would have little time to spend on this particular automation project. In my experience test automation projects work best if there is a dedicated engineer working exclusively on automation. Additional, dedicated staffing was not available.

I evaluated several automation tools, some open source, others proprietary: selenium, watir, twill, web replay, and squish. Somewhat on the merits of an O'Reilly article (http://oreilly.com/catalog/9780596527808/), I felt selenium and twill looked to be the best choices for Playlist: both were proven automation tools, free, and there was ample support available. But for meeting the criteria I outlined above, twill was the tool to work with first. My department would go on to build an excellent selenium test suite, but that would require more time and maintenance. I used twill to build a quick and dirty site monitor.

Details

Setting up python and twill

All work was done on WinXP Pro:

1. First I installed the latest Python release from here: http://www.python.org/download/releases/
2. Next I installed the setup tools from here: http://pypi.python.org/pypi/setuptools.
3. Last I installed twill by running the command "easy_install twill"

You can run all twill commands from the shell command line. However, I created a file(using Notepad++) which contained all twill commands. See below for details.

Scripting the site

This is straight forward:

1. Define the list of pages I wanted to check. I knew from Google Analytics the thirty or so pages at Playlist that received the majority of traffic.
2. Use the twill **go** and **code** commands to navigate to each page and check it's http status. For example, the script starts with the following:

```
go http://www.playlist.com
code 200
```

3. The script checks for an http status code of 200 (success, ok). If either the navigation or http status code check fails, the script aborts.
4. Proceed to next page.

Because twill does not interact with javascript, I could not use twill to run any music searches. However, I still used twill to navigate to the search page to ensure it was still up. See the script a the end of this document for the particulars.

The harness and running the script

While creating scripted test cases for a web site is relatively easy, setting up a test harness to schedule, run, and report results can be time consuming. I side stepped all this by simply using the task scheduler built into the Windows operating system. I scheduled a task to run a batch file that called the twill script every five(5) minutes. The script ran on an extra notebook computer I had, a test machine I used for secondary testing tasks. The batch file did a few house cleaning tasks, but existed mostly to set up and break down the twill script.

Checking for success or failure was decidedly low tech: since I was usually at my desk all day, I would simply glance at the machine running the twill scripts throughout the day. If the script failed, the twill console would be open and report it as such. Otherwise, if the script ran successfully, the twill console would close.

Next steps

Since the twill site check script described above was useful only when I was in the office to monitor the results, the script was of limited value. My next steps are to integrate blat into the Windows batch file. Blat is a Windows command line tool that emails the contents of a file via SMTP. This would permit me to send the result of a twill run to an email account.

I also want to modify the script to make the domain under test a variable, so I can reuse the same script on our internal sites, and simply pass in another domain as an parameter, e.g. “qatest-site.playlist.com”.

ROI

How often did my twill script detect a production web site problem ahead of the other reporting methods? I noted three times in a seven month period. This is not very many instances of error detection, but twill was simply one of several monitoring systems in place at Playlist. In addition, the investment time to build, deploy, and maintain the script was only several hours, a very small investment.

Going forward, as the both the qa department and the rest of Playlist engineering grows, we’ll retire this twill script and implement more robust, industrial production site monitoring. But for one overworked qa engineer/manager, this quick, easy, but useful twill script is the right tool at the right time.

Reference

Python

http://www.python.org/

Twill

http://twill.idyll.org/

Blat

http://www.blat.net/

O'Reilly article on twill and selenium

http://oreilly.com/catalog/9780596527808/

Script code

<start>

```
#check the primary pages at playlist.com, confirm HTTP code 200, then move
on to next page

go http://www.playlist.com
code 200
#check for usual page components
find http://static.pplaylist.com/players
find user
find login
find register
find /playlist/
find "Browse all playlists"
#find "Browse all members"
#find ringtones

#check ads
find doubleclick

#check the links on page; ads often fail, as do takeover artist searches
extend_with check_links
config check_links.only_collect_bad_links +
check_links

#search for Madonna and submit.
fv 3 searchfor Madonna
submit

#expect error return since twill doesn't work with javascript. But still
can check page components
code 200
find pageLoadError
find doubleclick
find searchplayer

#check account sign up page
go "http://www.playlist.com/user/register"
code 200

#check login page
go "http://www.playlist.com/user/login"
code 200

#check help page
go "http://www.playlist.com/help"
code 200

#check forums
go "http://www.playlist.com/forum"
```

```
code 200

go "http://www.playlist.com/forum/myspace"
code 200

go "http://www.playlist.com/forum/support"
code 200

go "http://www.playlist.com/forum/css_code"
code 200

go "http://www.playlist.com/forum/general"
code 200

#check the rest of the site

go "http://www.playlist.com/playlist"
code 200

go "http://www.playlist.com/help/contact"
code 200

go "http://www.playlist.com/about/terms"
code 200

go "http://www.playlist.com/about/privacy"
code 200

go "http://www.playlist.com/about"
code 200

go "http://www.playlist.com/about/webmasters"
code 200

go "http://www.playlist.com/about/copyright"
code 200

go "http://www.playlist.com/about/advertising"
code 200
<finish>
```

Batch file

This is the Windows batch file executed by the task scheduler.

```
cd \
cd C:\Playlist\Auto_and_scripting\Twill
twill-sh "twill site check.txt"
sleep 30
```

Good luck!