[P2PBrowser group] Code Architechture

Steffan Norberhuis Quinten Stokkink

12 June 2013

1 Introduction

Our projects consists of two seperate extensions of Tribler. Both use wxWebview to browse the internet. The first extension is the Eternal Webpage extension. This extension will allow a user to easily browse the web using peer-to-peer-swarms and also add new pages to the swarm. The second extension is the TUPT extension. This extension allows you to browse websites and if a movie is detected on the website, then you can begin streaming this movie with one click of a button.

The only common part between the two extensions is the WebBrowser class. After we finished the Eternal Webpages extension, we started working on the TUPT extension. We transformed the WebBrowser class and stripped all functionality specific for the EternalWebpages extension that was no longer needed. All needed functionality for the TUPT extension to the WebBrowser class has been added in a generic way. This generic approach was much harder for the WebBrowser, because we changed the way how WebBrowser loads pages.

2 Eternal Webpages diagrams

The Eternal Webpages package is a Proof Of Concept package no longer undergoing active development. Its main purpose is to download and bundle resources referenced by a webpage in such a way that they can be fetched offline. To accomplish this, the webbrowser has 2 modes for viewing websites: one by browsing the internet and one by downloading P2P cached copies of websites (called internet viewmode and swarm viewmode respectively). This is shown in Figure 1. The package is referenced from the main Tribler projects web browser for every resource it ecounters while loading a webpage. This is shown in Figure 2. The full class diagram of the package is included in Figure 3.

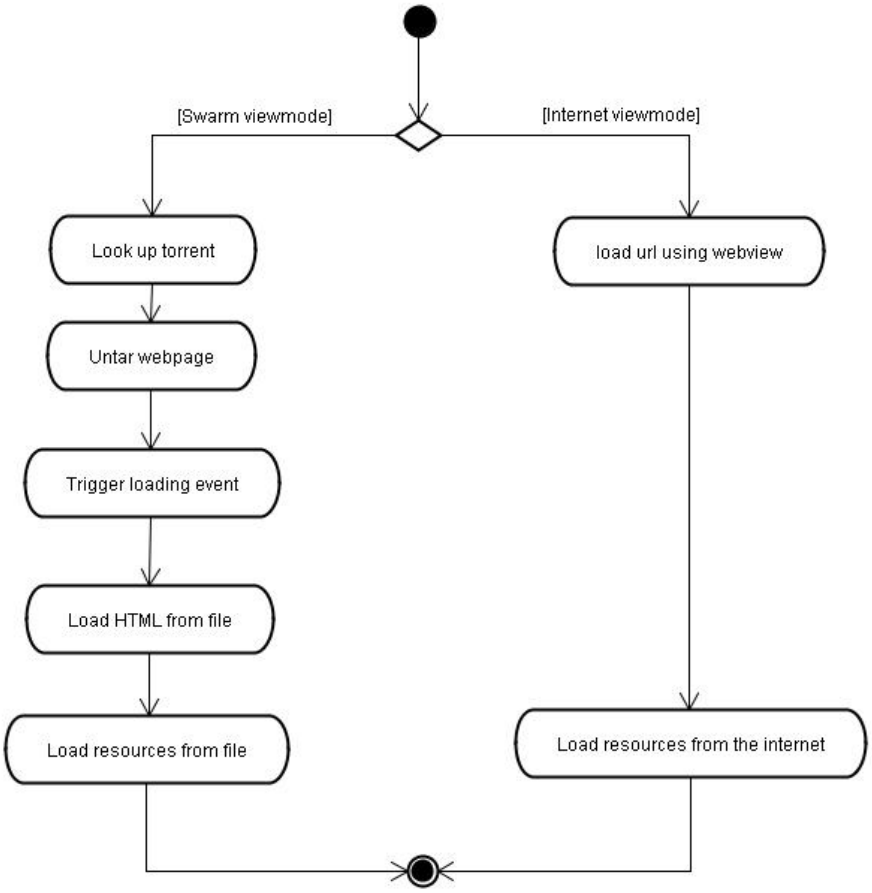

Figure 1: Activity diagram of viewmode based webbrowsing

3 TUPT diagrams

4 Tests

Our package comes with a set of python 'unittest' files. This section maps these unit-tests and explains how to run them. Some unit-tests cannot be run however due to the separation from the main Tribler project. There are two unit-tests for which this is an issue. As with the main project, all the tests are written for Python 2.7 . We also want to note that some unit-tests require an active internet connection.

4.1 Required packages

To run our tests we use Ubuntu machines. In Table 1 are the required 3rd party python packages and their Ubuntu package names required to run the unit-tests.

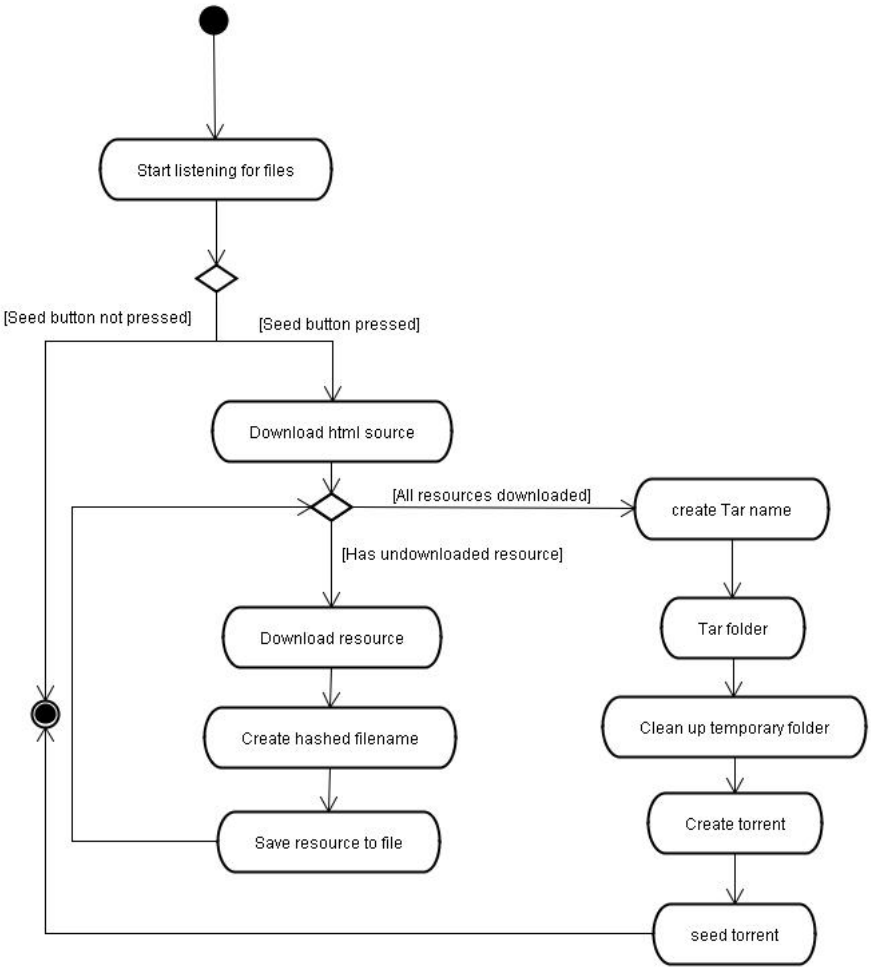

Figure 2: Activity diagram of seeding mode webbrowsing

4.2 Non-runnable tests

There are two unit-tests we cannot deliver runnable. Table Table 2 lays out the tests and the rationale behind why they are not runnable. Note all of the paths in the file names are offset from the *Tribler/Test/TUPT/* folder. These tests rely so heavily on core Tribler functionality, stubbing the Tribler classes would lead to unacceptable overlap of the stubs with the actual classes.

4.3 Runnable tests

The following tests in Table 3 can be run normally (provided the 3rd party packages were properly installed) from the terminal. The file names are offset from the *Tribler/Test/* folder.

Table 1: Required packages

Name	Ubuntu package
Yapsy	python-yapsy
wxPython	python-wxgtk2.8
IMDbPY	python-imdbpy
BeautifulSoup	python-bs4

Table 2: Non-runnable tests

File	Rationale
test_TUPTControl.py	Requires a development build of wxPython (2.9.4.0) and relies heavily on core Tribler functionality.
Channels/test_ChannelControl.py	Relies heavily on core Tribler functionality.

4.4 Test output

This subsection contains the console output for all of our tests. The raw console output is shown for each of the tests in Table 4

Table 4: Test output

File	**Output**

TUPT/test_TUPTControl	`Finding files... done.`
	`Importing test modules ... done.`
	
	`----------`
	`Ran 13 tests in 0.002s`
	
	`OK`

Table 4 – continued from previous page

File	Output
TUPT/Channels/test_ChannelControl	Finding files... done. Importing test modules ... done. ---------- Ran 5 tests in 0.002s OK
PluginManager/test_PluginManager	Finding files... done. Importing test modules ... done. ---------- Ran 3 tests in 0.031s OK
TUPT/Parser/test_ParserControl	Finding files... done. Importing test modules ... done. ---------- Ran 6 tests in 0.001s OK
TUPT/TorrentFinder/test_SortedTorrentList	Finding files... done. Importing test modules ... done. ---------- Ran 6 tests in 0.004s OK

Table 4 – continued from previous page

File	Output

TUPT/Matcher/test_MatcherControl	`Finding files... done.`
	`Importing test modules ... done.`
	
	`----------`
	`Ran 2 tests in 2.541s`
	
	`OK`

TUPT/Matcher/
test_TestTheMovieDBMatcherPlugin
	`Finding files... done.`
	`Importing test modules ... done.`
	
	`----------`
	`Ran 3 tests in 2.274s`
	
	`OK`

TUPT/Parser/test_IMDbParserPlugin	`Finding files... done.`
	`Importing test modules ... done.`
	
	`----------`
	`Ran 7 tests in 4.631s`
	
	`OK`

TUPT/TorrentFinder/
test_KatPhTorrentFinderPlugin
	`Finding files... done.`
	`Importing test modules ... done.`
	
	`----------`
	`Ran 3 tests in 0.970s`
	
	`OK`

Table 4 – continued from previous page

File	Output
TUPT/TorrentFinder/ test_TorrentFinderControl	Finding files... done. Importing test modules ... done. ---------- Ran 10 tests in 0.007s OK

Table 3: Runnable tests

PluginManager/test_PluginManager.py
TUPT/Parser/test_ParserControl
TUPT/TorrentFinder/test_SortedTorrentList
TUPT/Matcher/test_MatcherControl
TUPT/Matcher/test_TestTheMovieDBMatcherPlugin
TUPT/Parser/test_IMDbParserPlugin
TUPT/TorrentFinder/test_KatPhTorrentFinderPlugin
TUPT/TorrentFinder/test_TorrentFinderControl

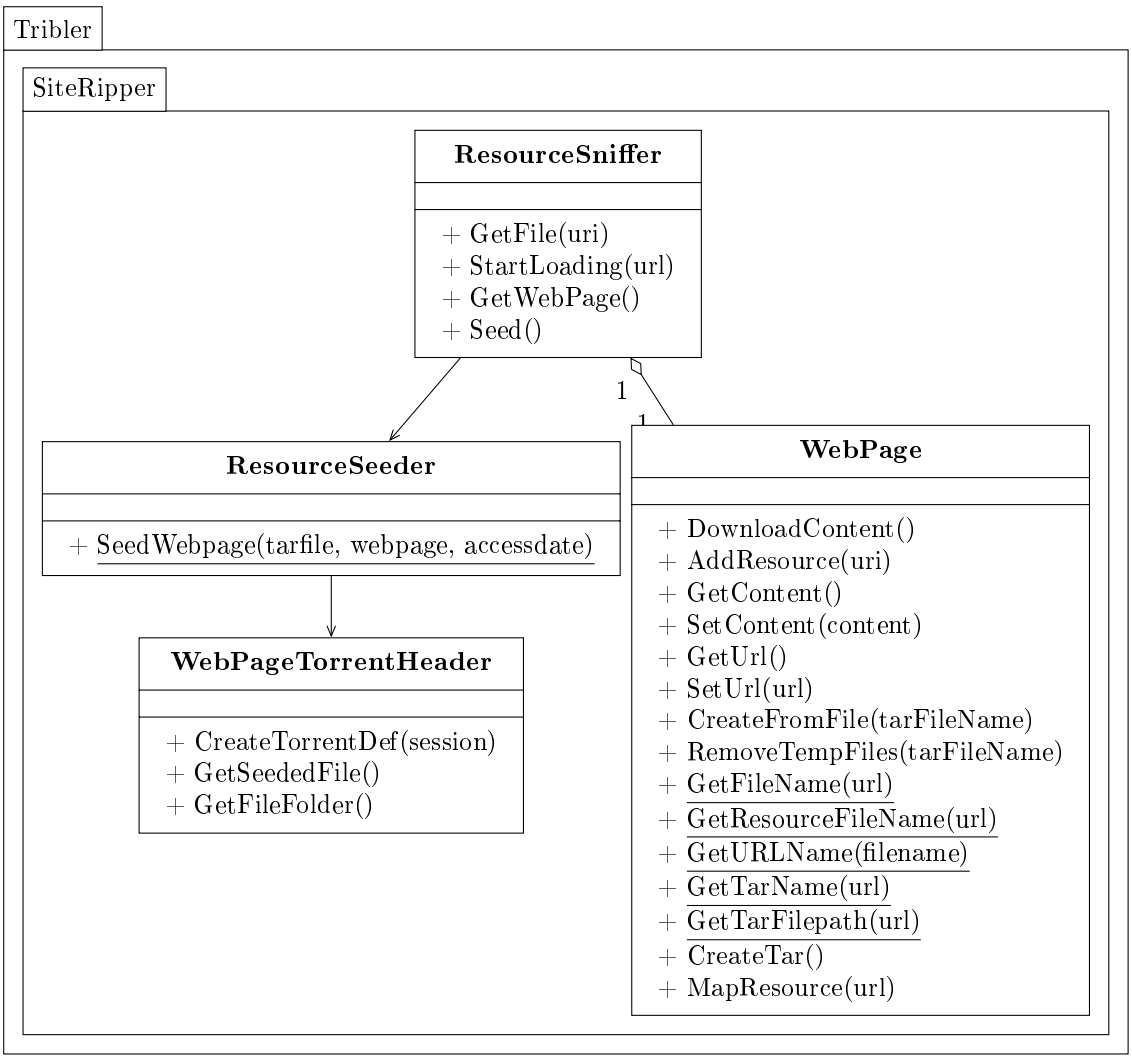

Figure 3: Class diagram for Eternal Webpages