

# websitary

Thomas Link

24. Mai 2008

## Inhaltsverzeichnis

<b>1 DESCRIPTION:</b>	<b>2</b>
<b>2 FEATURES/PROBLEMS:</b>	<b>2</b>
2.1 Caveat . . . . .	3
<b>3 SYNOPSIS:</b>	<b>3</b>
3.1 Usage . . . . .	3
3.2 Configuration . . . . .	4
3.2.1 default 'PROFILE1', 'PROFILE2' ... . . . .	4
3.2.2 diff 'CMD "%s" "%s" . . . . .	4
3.2.3 diffprocess lambda { text  ...} . . . . .	5
3.2.4 download 'CMD "%s" . . . . .	5
3.2.5 downloadprocess lambda { text  ...} . . . . .	5
3.2.6 edit 'CMD "%s" . . . . .	5
3.2.7 option TYPE, OPTION => VALUE . . . . .	5
3.2.8 global OPTION => VALUE . . . . .	6
3.2.9 output_format FORMAT, output_format [FOR- MAT1, FORMAT2, ...] . . . . .	6
3.2.10 set OPTION => VALUE; set TYPE, OPTION => VALUE; unset OPTIONS . . . . .	7
3.2.11 source URL(S), [OPTIONS] . . . . .	7
3.2.12 view 'CMD "%s" . . . . .	10
3.3 Shortcuts for use with :use, :download and other options . . .	10
3.4 Example configuration file for demonstration purposes . . . .	11
3.5 Commands for use with the -e command-line option . . . . .	14
<b>4 TIPS:</b>	<b>15</b>
4.1 Ruby . . . . .	15
4.2 Cygwin . . . . .	15
4.3 Windows . . . . .	15
<b>5 REQUIREMENTS:</b>	<b>15</b>

<b>6</b>	<b>INSTALL:</b>	<b>16</b>
6.1	Use rubygems . . . . .	16
6.2	Use the zip . . . . .	17
6.3	Initial Configuration . . . . .	17

**7 LICENSE: 18**

websitary by Thomas Link <http://rubyforge.org/projects/websitary/>

This script monitors webpages, rss feeds, podcasts etc. and reports what's new. For many tasks, it reuses other programs to do the actual work. By default, it works on an ASCII basis, i.e. with the output of text-based webbrowsers. With the help of some friends, it works also with HTML.

**1 DESCRIPTION:**

websitary (formerly known as websitary with an extra "i") monitors webpages, rss feeds, podcasts etc. It reuses other programs (w3m, diff etc.) to do most of the actual work. By default, it works on an ASCII basis, i.e. with the output of text-based webbrowsers like w3m (or lynx, links etc.) as the output can easily be post-processed. It can also work with HTML and highlight new items. This script was originally planned as a ruby-based websec replacement.

By default, this script will use w3m to dump HTML pages and then run diff over the current page and the previous backup. Some pages are better viewed with lynx or links. Downloaded documents (HTML or ASCII) can be post-processed (e.g., filtered through some ruby block that extracts elements via hpricot and the like). Please see the configuration options below to find out how to change this globally or for a single source.

This user manual is also available as PDF.

**2 FEATURES/PROBLEMS:**

- Handle webpages, rss feeds (optionally save attachments in podcasts etc.)
- Compare webpages with previous backups
- Display differences between the current version and the backup
- Provide hooks to post-process the downloaded documents and the diff
- Display a one-page report summarizing all news
- Automatically open the report in your favourite web-browser
- Experimental: Download webpages on defined intervalls and generate incremental diffs.

ISSUES, TODO:

- With HTML output, changes are presented on one single page, which means that pages with different encodings cause problems.
- Improved support for robots.txt (test it)
- The use of :website\_below and :website is hardly tested (please report errors).
- download => :body\_html tries to rewrite references (a, img) which may fail on certain kind of urls (please report errors).
- When using :body\_html for download, it may happen that some JavaScript code is stripped, which breaks some JavaScript-generated links.
- The -log command line will create a new instance of the logger and thus reset any previous options related to the logging level.

NOTE: The script was previously called websituary but was renamed (from 0.2 on) to websitary (without the superfluous i).

## 2.1 Caveat

The script also includes experimental support for monitoring whole websites. Basically, this script supports robots.txt directives (see requirements) but this is hardly tested and may not work in some cases.

While it is okay for your own websites to ignore robots.txt, it is not for others. Please make sure that the webpages you run this program on allow such a use. Some webpages disallow the use of any automatic downloader or offline reader in their user agreements.

## 3 SYNOPSIS:

### 3.1 Usage

Example:

```
# Run "profile"
websitary profile

# Edit "~/.websitary/profile.rb"
websitary --edit=profile
```

```

# View the latest report
websitary -ereview

# Refetch all sources regardless of :days and :hours restrictions
websitary -signore_age=true

# Create html and rss reports for my websites
websitary -fhtml,rss mysites

# Add an url to the quicklist profile
websitary -eadd http://www.example.com

```

For example output see:

- html
- rss
- text

## 3.2 Configuration

Profiles are plain ruby files (with the '.rb' suffix) stored in ~/.websitary/.

The profile "config" (~/.websitary/config.rb) is always loaded if available.

There are two special profile names:

-

Read URLs from STDIN.

\_\_END\_\_ Read the profile contained in the script source after the END line.

### 3.2.1 default 'PROFILE1', 'PROFILE2' ...

Set the default profile(s). The default is: quicklist

Example:

```
default 'my_profile'
```

### 3.2.2 diff 'CMD "%s" "%s"'

Use this shell command to make the diff. %s %s will be replaced with the old and new filename.

diff is used by default.

### 3.2.3 diffprocess lambda {|text| ...}

Use this ruby snippet to post-process the diff.

### 3.2.4 download 'CMD "%s"'

Use this shell command to download a page. %s will be replaced with the url.

w3m is used by default.

Example:

```
download 'lynx -dump "%s"'
```

### 3.2.5 downloadprocess lambda {|text| ...}

Use this ruby snippet to post-process what was downloaded. Return the new text.

### 3.2.6 edit 'CMD "%s"'

Use this shell command to edit a profile. %s will be replaced with the filename.

vi is used by default.

Example:

```
edit 'gvim "%s"&'
```

### 3.2.7 option TYPE, OPTION => VALUE

Set a global option.

TYPE can be one of:

**:diff** Generate a diff

**:diffprocess** Post-process a diff (if necessary)

**:format** Format the diff for output

**:download** Download webpages

**:downloadprocess** Post-process downloaded webpages

**:page** The **:format** field defines the format of the final report. Here VALUE is a format string that takes 3 variables as arguments: report title, toc, contents.

**:global** Set a "global" option.

DOWNLOAD is a symbol  
VALUE is either a format string or a block of code (of class Proc).

Example:

```
set :download, :foo => lambda {|url| get_url(url)}
```

### 3.2.8 global OPTION => VALUE

This is the same as a option :global, OPTION => VALUE.

Known global options:

`:canonic_filename => BLOCK(FILENAME)` Rewrite filenames as they are stored in the mtime register. This may

useful if you want to use the same repository on several computers with in different locations etc.

`:encoding => OUTPUT_DOCUMENT_ENCODING` The default is 'ISO-8859-1'.

`:downloadhtml => SHORTCUT` The default shortcut for downloading plain HTML.

`:file_url => BLOCK(FILENAME)` Rewrite a filename as it is used for creating file urls to local

copies in the output. This may useful if you want to use the same repository on several computers with in different locations etc.

`:filename_size => N` The max filename size. If a filename becomes longer, md5 encoding will

be used for local copies in the cache.

`:toggle_body => BOOLEAN` If true, make a news body collapsable on mouse-clicks (sort of).

`:proxy => STRING, :proxy => ARRAY` The proxy. (currently only supported by mechanize)

`:user_agent => STRING` Set the user agent (only for certain queries).

### 3.2.9 output\_format FORMAT, output\_format [FORMAT1, FORMAT2, ...]

Set the output format. Format can be one of:

- html
- text, txt (this only works with text based downloaders)

- rss (prove of concept only; it requires :rss[:url] to be set to the url, where the rss feed will be published, using the option :rss, :url => URL configuration command; you either have to use a text-based downloader or include :rss\_format => 'html' to the url options)

### 3.2.10 set OPTION => VALUE; set TYPE, OPTION => VALUE; unset OPTIONS

(Un)Set an option for the following source commands.

Example:

```
set :download, :foo => lambda {|url| get_url(url)}
set :days => 7, sort => true
unset :days, :sort
```

### 3.2.11 source URL(S), [OPTIONS]

Options

:cols => FROM..TO Use only these columns from the output (used after applying the :lines option)

:depth => INTEGER In conjunction with a :website type of :download option, fetch url up to this depth.

:diff => ‘‘CMD‘‘, :diff => SHORTCUT Use this command to make the diff for this page. Possible values for

SHORTCUT are: :webdiff (useful in conjunction with :download => :curl, :wget, or :body\_html), :websec\_webdiff (use websec’s webdiff tool), :body\_html, :website\_below, :website and :openuri are synonyms for :webdiff. NOTE: Since version 0.3, :webdiff is mapped to websitary’s own htmdiff class (which can also be used as stand-alone script). Before 0.3, websitary used websec’s webdiff script, which is now mapped to :websec\_webdiff.

:diffprocess => lambda {|text| ...} Use this ruby snippet to post-process this diff

:download => ‘‘CMD‘‘, :download => SHORTCUT Use this command to download this page. For possible values for SHORTCUT see the section on shortcuts below.

`:downloadprocess => lambda {|text| ...}` Use this ruby snippet to post-process what was downloaded. This is the place where, e.g., `hpricot` can be used to extract certain elements from the HTML code. Example: `lambda {|text| Hpricot(text).at('div#content').inner_html}`

`:format => 'FORMAT %s STRING'`, `:format => SHORTCUT` The format string for the diff text. The default (the `:diff` shortcut) wraps the output in `pre` tags. `:webdiff`, `:body_html`, `:website_below`, `:website`, and `:openuri` will simply add a newline character.

`:iconv => ENCODING` If set, use `iconv` to convert the page body into the summary's document encoding (see the 'global' section). Websitary currently isn't able to automatically determine and convert encodings.

`:timeout => SECONDS` When using `openuri`, download the page with a timeout.

`:hours => HOURS`, `:days => DAYS` Don't download the file unless it's older than that

`:days_of_month => DAY..DAY`, `:wdays => DAY..DAY` Download only once per month within a certain range of days (e.g., `15..31` ... Check once after the 15th). The argument can also be an array (e.g, `[1, 15]`) or an integer.

`:days_of_week => DAY..DAY`, `:mdays => DAY..DAY` Download only once per week within a certain range of days (e.g., `1..2` ... Check once on monday or tuesday; `sunday = 0`). The argument can also be an array (e.g, `[1, 15]`) or an integer.

`:daily => true` Download only once a day.

`:ignore_age => true` Ignore any `:days` and `:hours` settings. This is useful in some cases when set on the command line.

`:lines => FROM..TO` Use only these lines from the output

`:match => REGEXP` When recursively walking a website, follow only links that match this regexp.

`:rss_rewrite_enclosed_urls => true` If true, replace urls in the rss feed item description pointing to the enclosure with a file url pointing to the local copy

`:rss_enclosure => true|'DIRECTORY'` If true, save rss feed enclosures in  
 “~/websitary/attachments/RSS\_FEED\_NAME/“. If a string, use this as destination directory. Only enclosures of new items will be saved – i.e. when downloading a feed for the first time, no enclosures will be saved.

`:rss_find_enclosure => BLOCK` Certain RSS-feeds embed enclosures in the description. Use this option  
 to scan the description (a Hpricot document) for an URL that is then saved as enclosure if the `:rss_enclosure` option is set. Example: `source 'http://www.example.com/rss', :title => 'Example', :use => :rss, :rss_enclosure => true, :rss_find_enclosure => lambda {|item, doc| (doc / 'img').map {|e| e['src']}[0]}`

`:rss_format (default: 'plain_text')` When output format is `:rss`, create rss item descriptions as plain text.

`:rss_format_local_copy => FORMAT_STRING | BLOCK` By default a hyper-text reference to the local copy of an RSS  
 enclosure is added to entry. Sometimes you may want to display something inline (e.g. an image). You can then use this option to define a format string (one field = the local copy's file url).

`:show_initial => true` Include initial copies in the report (may not always work properly).  
 This can also be set as a global option.

`:sleep => SECS` Wait SECS seconds (float or integer) before downloading the page.

`:sort => true, :sort => lambda {|a,b| ...}` Sort lines in output

`:strip => true` Strip empty lines

`:title => 'TEXT'` Display TEXT instead of URL

`:use => SYMBOL` Use SYMBOL for any other option. I.e. `<tt>:download => :body_html`  
`:diff => :webdiff` can be abbreviated as `:use => :body_html` (because for `:diff` `:body_html` is a synonym for `:webdiff`).

The order of age constraints is: :hours > :daily > :wdays > :mdays > :days > :months. I.e. if :wdays is set, :mdays, :days, or :months are ignored.

### 3.2.12 view 'CMD "%s"'

Use this shell command to view the output (usually a HTML file). %s will be replaced with the filename.

w3m is used by default.

Example:

```
view 'gnome-open "%s"' # Gnome Desktop
view 'kfmclient "%s"' # KDE
view 'cygstart "%s"' # Cygwin
view 'start "%s"' # Windows
view 'firefox "%s"'
```

## 3.3 Shortcuts for use with :use, :download and other options

**:w3m** Use w3m for downloading the source. Use diff for generating diffs.

**:lynx** Use lynx for downloading the source. Use diff for generating diffs.

Lynx doesn't try to recreate the layout of a page like w3m or links do. As a result the output IMHO sometimes deviates from the original design but is better suited for being post-processed in some situation.

**:links** Use links for downloading the source. Use diff for generating diffs.

**:curl** Use curl for downloading the source. Use webdiff for generating diffs.

**:wget** Use wget for downloading the source. Use webdiff for generating diffs.

**:openuri** Use open-uri for downloading the source. Use webdiff for generating

diffs. This doesn't handle cookies and the like.

**:mechanize** Use mechanize (must be installed) for downloading the source.

Use

webdiff for generating diffs. This calls the URL's :mechanize property (a lambda that takes 3 arguments: URL, agent, page => HTML as string) to post-process the page (or if not available, use the page body's HTML).

**:text** This requires hpricot to be installed. Use open-uri for downloading and hpricot for converting HTML to plain text. This still requires diff as external helper.

`:body_html` This requires hpricot to be installed. Use open-uri for downloading the source, use only the body. Use webdiff for generating diffs. Try to rewrite references (a, img) so that they point to the webpage. By default, this will also strip tags like script, form, object ...

`:website` Use `:body_html` to download the source. Follow all links referring to the same host with the same file suffix. Use webdiff for generating diff.

`:website_below` Use `:body_html` to download the source. Follow all links referring to the same host and a file below the top directory with the same file suffix. Use webdiff for generating diff.

`:website_txt` Use `:website` to download the source but convert the output to plain text.

`:website_txt_below` Use `:website_below` to download the source but convert the output to plain text.

`:rss` Download an rss feed, show changed items.

`:opml` Experimental. Download the rss feeds registered in opml. No support for atom yet.

`:img` Download an image and display it in the output if it has changed (according to diff). You can use hpricot to extract an image from a HTML source. Example:

Any shortcuts relying on `:body_html` will also try to rewrite any references so that the links point to the webpage.

### 3.4 Example configuration file for demonstration purposes

```
# Daily
set :days => 1

# Use lynx instead of the default downloader (w3m).
source 'http://www.example.com', :days => 7, :download => :lynx
```

```

# Use the HTML body and process via webdiff.
source 'http://www.example.com', :use => :body_html,
      :downloadprocess => lambda {|text| Hpricot(text).at('div#content').inner_html}

# Download a podcast
source 'http://www.example.com/podcast.xml', :title => 'Podcast',
      :use => :rss,
      :rss_enclosure => '/home/me/podcasts/example'

# Check a rss feed.
source 'http://www.example.com/news.xml', :title => 'News', :use => :rss

# Get rss feed info from an opml file (EXPERIMENTAL).
# @cfgdir is most likely '~/websitary'.
source File.join(@cfgdir, 'news.opml'), :use => :opml

# Weekly
set :days => 7

# Consider the page body only from the 10th line downwards.
source 'http://www.example.com', :lines => 10..-1, :title => 'My Page'

# Bi-weekly
set :days => 14

# Use these urls with the default options.
source <<URLS
http://www.example.com
http://www.example.com/page.html
URLS

```

```

# Make HTML diffs and highlight occurrences of a word
source 'http://www.example.com',
  :title => 'Example',
  :use => :body_html,
  :diffprocess => highlighter(/word/i)

# Download the whole website below this path (only pages with
# html-suffix), wait 30 secs between downloads.
# Download only php and html pages
# Follow links 2 levels deep
source 'http://www.example.com/foo/bar.html',
  :title => 'Example -- Bar',
  :use => :website_below, :sleep => 30,
  :match => /\. (php|html)\b/, :depth => 2

# Download images from some kind of daily-image site (check the user
# agreement first, if this is allowed). This may require some ruby
# hacking in order to extract the right url.
source 'http://www.example.com/daily_image/', :title => 'Daily Image',
  :use => :img,
  :download => lambda {|url|
    rv = nil
    # Read the HTML.
    html = open(url) {|io| io.read}
    # This check is probably unnecessary as the failure to read
    # the HTML document would most likely result in an
    # exception.
    if html
      # Parse the HTML document.
      doc = Hpricot(html)
      # The following could actually be simplified using xpath
      # or css search expressions. This isn't the most elegant
      # solution but it works with any value of ALT.
      # This downloads the image 
      # Check all img tags in the HTML document.
      for e in doc.search(%{//img})
        # Is this the image we're looking for?
        if e['alt'] == "Current Image"
          # Make relative urls absolute
          img = rewrite_href(e['src'], url)
          # Get the actual image data
          rv = open(img, 'rb') {|io| io.read}
        end
      end
    end
  }

```

```

        # Exit the for loop
        break
    end
end
rv
end
}

```

```
unset :days
```

### 3.5 Commands for use with the -e command-line option

Most of these commands require you to name a profile on the command line. You can define default profiles with the “default“ configuration command.

If no command is given, “downdiff“ is executed.

**add** Add the URLs given on the command line to the quicklist profile.

ATTENTION: The following arguments on the command line are URLs, not profile names.

**aggregate** Retrieve information and save changes for later review.

**configuration** Show the fully qualified configuration of each source.

**downdiff** Download and show differences (DEFAULT)

**edit** Edit the profile given on the command line (use vi by default)

**latest** Show the latest copies of the sources from the profiles given on the command line.

**ls** List number of aggregated diffs.

**rebuild** Rebuild the latest report.

**review** Review the latest report (just show it with the browser)

**show** Show previously aggregated items. A typical use would be to periodically run in the background a command like `websitary -eaggregate` newsfeeds and then `websitary -eshow` newsfeeds to review the changes.

**unroll** Undo the latest fetch.

## 4 TIPS:

### 4.1 Ruby

The profiles are regular ruby sources that are evaluated in the context of the configuration object (`Websitary::Configuration`). Find out more about ruby at:

- <http://www.ruby-lang.org/en/documentation/>
- <http://www.ruby-doc.org/docs/ProgrammingRuby/> (especially the language chapter)

### 4.2 Cygwin

Mixing native Windows apps and cygwin apps can cause problems. The following settings (e.g. in `~/.websitary/config.rb`) can be used to use a native Windows editor and browser:

```
# Use the default Windows programs (as if double-clicked)
view '/usr/bin/cygstart "%s"'

# Translate the profile filename and edit it with a native Windows editor
edit 'notepad.exe $(cygpath -w -- "%s")'

# Rewrite cygwin filenames for use with a native Windows browser
option :global, :file_url => lambda {|f| f.sub(/\/cygdrive\/.+?\/.websitary\/\//,
''')}

```

### 4.3 Windows

Backslashes usually have to be escaped by backslashes – or use slashes. I.e. instead of `'c:\foo\bar'` write either `'c:\\foo\\bar'` or `'c:/foo/bar'`.

## 5 REQUIREMENTS:

`websitary` is a ruby-based application. You thus need a ruby interpreter.

It depends on how you use `websitary` whether you actually need the following libraries, applications.

By default this script expects the following applications to be present:

- diff
- vi (or some other editor)

and one of:

- w3m (default)
- lynx
- links

The use of `:websec_webdiff` as `:diff` application requires `websec` (or at Savannah) to be installed. By default, `websitary` uses its own `htmldiff` class/script, which is less well tested and may return inferior results in comparison with `websec`'s `webdiff`. In conjunction with `:body_html`, `:openuri`, or `:curl`, this will give you colored HTML diffs.

For downloading HTML, you need one of these:

- `open-uri` (should be part of ruby)
- `hpricot` (used e.g. by `:body_html`, `:website`, and `:website_below`)
- `curl`
- `wget`

The following ruby libraries are needed in conjunction with `:body_html` and `:website` related shortcuts:

- `hpricot` (parse HTML, use only the body etc.)
- `robot_rules.rb` for parsing `robots.txt`

I personally would suggest to choose the following setup:

- `w3m`
- `hpricot`
- `robot_rules.rb`

## 6 INSTALL:

### 6.1 Use rubygems

Run

```
gem install websitary
```

This will download the package and install it.

## 6.2 Use the zip

The zip contains a file `setup.rb` that does the work. Run

```
ruby setup.rb
```

## 6.3 Initial Configuration

Please check the requirements section above and get the extra libraries needed:

- `hpricot`
- `robot_rules.rb`

These could be installed by:

```
# Install hpricot
gem install hpricot

# Install robot_rules.rb
wget http://www.rubyquiz.com/quiz64_sols.zip
# Check the correct path to site_ruby first!
unzip -p quiz64_sols.zip "solutions/James Edward Gray II/robot_rules.rb"
> /lib/ruby/site_ruby/1.8/robot_rules.rb
rm quiz64_sols.zip
```

You might then want to create a profile `~/.websitary/config.rb` that is loaded on every run. In this profile you could set the default output viewer and profile editor, as well as a default profile.

Example:

```
# Load standard.rb if no profile is given on the command line.
default 'standard'

# Use cygwin's cygstart to view the output with the default HTML
# viewer
view '/usr/bin/cygstart "%s"'

# Use Windows gvim from cygwin ruby which is why we convert the path
# first
edit 'gvim $(cygpath -w -- "%s")'
```

Where these configuration files reside, may differ. If the environment variable \$HOME is defined, the default is \$HOME/.websitary/ unless one of the following directories exist, which will then be used instead:

- \$USERPROFILE/websitary (on Windows)
- SYSCONFDIR/websitary (where SYSCONFDIR usually is /etc but you can run ruby to find out more:  
`ruby -e 'p Config::CONFIG['sysconfdir']'`)

If neither directory exists and no \$HOME variable is defined, the current directory will be used.

Now check out the configuration commands in the Synopsis section.

## 7 LICENSE:

websitary Webpage Monitor Copyright (C) 2007 Thomas Link

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA