21.1. webbrowser — Convenient Web-browser controller

Source code: Lib/webbrowser.py

The webbrowser module provides a high-level interface to allow displaying Web-based documents to users. Under most circumstances, simply calling the open() function from this module will do the right thing.

Under Unix, graphical browsers are preferred under X11, but text-mode browsers will be used if graphical browsers are not available or an X11 display isn't available. If text-mode browsers are used, the calling process will block until the user exits the browser.

If the environment variable BROWSER exists, it is interpreted as the os.pathsep-separated list of browsers to try ahead of the platform defaults. When the value of a list part contains the string %s, then it is interpreted as a literal browser command line to be used with the argument URL substituted for %s; if the part does not contain %s, it is simply interpreted as the name of the browser to launch. [1]

For non-Unix platforms, or when a remote browser is available on Unix, the controlling process will not wait for the user to finish with the browser, but allow the remote browser to maintain its own windows on the display. If remote browsers are not available on Unix, the controlling process will launch a new browser and wait.

The script **webbrowser** can be used as a command-line interface for the module. It accepts an URL as the argument. It accepts the following optional parameters: -n opens the URL in a new browser window, if possible; -t opens the URL in a new browser page ("tab"). The options are, naturally, mutually exclusive. Usage example:

```
python -m webbrowser -t "http://www.python.org"
```

The following exception is defined:

exception webbrowser.**Error**

Exception raised when a browser control error occurs.

The following functions are defined:

webbrowser.**open**(*url*, *new=0*, *autoraise=True*)

Display *url* using the default browser. If *new* is 0, the *url* is opened in the same browser window if possible. If *new* is 1, a new browser window is opened if possible. If *new* is 2, a

new browser page (“tab”) is opened if possible. If *autoraise* is `True`, the window is raised if possible (note that under many window managers this will occur regardless of the setting of this variable).

Note that on some platforms, trying to open a filename using this function, may work and start the operating system’s associated program. However, this is neither supported nor portable.

`webbrowser.`**`open_new`**(*url*)

Open *url* in a new window of the default browser, if possible, otherwise, open *url* in the only browser window.

`webbrowser.`**`open_new_tab`**(*url*)

Open *url* in a new page (“tab”) of the default browser, if possible, otherwise equivalent to `open_new()`.

`webbrowser.`**`get`**(*using=None*)

Return a controller object for the browser type *using*. If *using* is `None`, return a controller for a default browser appropriate to the caller’s environment.

`webbrowser.`**`register`**(*name*, *constructor*, *instance=None*)

Register the browser type *name*. Once a browser type is registered, the `get()` function can return a controller for that browser type. If *instance* is not provided, or is `None`, *constructor* will be called without parameters to create an instance when needed. If *instance* is provided, *constructor* will never be called, and may be `None`.

This entry point is only useful if you plan to either set the `BROWSER` variable or call `get()` with a nonempty argument matching the name of a handler you declare.

A number of browser types are predefined. This table gives the type names that may be passed to the `get()` function and the corresponding instantiations for the controller classes, all defined in this module.

Type Name	Class Name	Notes
`'mozilla'`	`Mozilla('mozilla')`	
`'firefox'`	`Mozilla('mozilla')`	
`'netscape'`	`Mozilla('netscape')`	
`'galeon'`	`Galeon('galeon')`	
`'epiphany'`	`Galeon('epiphany')`	
`'skipstone'`	`BackgroundBrowser('skipstone')`	
`'kfmclient'`	`Konqueror()`	(1)
`'konqueror'`	`Konqueror()`	(1)
`'kfm'`	`Konqueror()`	(1)

Type Name	Class Name	Notes
`'mosaic'`	`BackgroundBrowser('mosaic')`	
`'opera'`	`Opera()`	
`'grail'`	`Grail()`	
`'links'`	`GenericBrowser('links')`	
`'elinks'`	`Elinks('elinks')`	
`'lynx'`	`GenericBrowser('lynx')`	
`'w3m'`	`GenericBrowser('w3m')`	
`'windows-default'`	`WindowsDefault`	(2)
`'macosx'`	`MacOSX('default')`	(3)
`'safari'`	`MacOSX('safari')`	(3)
`'google-chrome'`	`Chrome('google-chrome')`	
`'chrome'`	`Chrome('chrome')`	
`'chromium'`	`Chromium('chromium')`	
`'chromium-browser'`	`Chromium('chromium-browser')`	

Notes:

1. “Konqueror” is the file manager for the KDE desktop environment for Unix, and only makes sense to use if KDE is running. Some way of reliably detecting KDE would be nice; the `KDEDIR` variable is not sufficient. Note also that the name “kfm” is used even when using the **konqueror** command with KDE 2 — the implementation selects the best strategy for running Konqueror.
2. Only on Windows platforms.
3. Only on Mac OS X platform.

New in version 3.3: Support for Chrome/Chromium has been added.

Here are some simple examples:

```
url = 'http://docs.python.org/'

# Open URL in a new tab, if a browser window is already open.
webbrowser.open_new_tab(url)

# Open URL in new window, raising the window if possible.
webbrowser.open_new(url)
```

21.1.1. Browser Controller Objects

Browser controllers provide these methods which parallel three of the module-level convenience functions:

`controller.`**`open`**(*url*, *new=0*, *autoraise=True*)

Display *url* using the browser handled by this controller. If *new* is 1, a new browser window is opened if possible. If *new* is 2, a new browser page (“tab”) is opened if possible.

`controller.`**`open_new`**(*url*)

Open *url* in a new window of the browser handled by this controller, if possible, otherwise, open *url* in the only browser window. Alias `open_new()`.

`controller.`**`open_new_tab`**(*url*)

Open *url* in a new page (“tab”) of the browser handled by this controller, if possible, otherwise equivalent to `open_new()`.

Footnotes

[1] Executables named here without a full path will be searched in the directories given in the `PATH` environment variable.