urllib.robotparser — Internet Spider Access Control

Purpose: Parse `robots.txt` file used to control Internet spiders

`robotparser` implements a parser for the `robots.txt` file format, including a function that checks if a given user agent can access a resource. It is intended for use in well-behaved spiders, or other crawler applications that need to either be throttled or otherwise restricted.

robots.txt

The `robots.txt` file format is a simple text-based access control system for computer programs that automatically access web resources (“spiders”, “crawlers”, etc.). The file is made up of records that specify the user agent identifier for the program followed by a list of URLs (or URL prefixes) the agent may not access.

This is the `robots.txt` file for `https://pymotw.com/`:

```
# robots.txt


Sitemap: https://pymotw.com/sitemap.xml
User-agent: *
Disallow: /admin/
Disallow: /downloads/
Disallow: /media/
Disallow: /static/
Disallow: /codehosting/
```

It prevents access to some of the parts of the site that are expensive to compute and would overload the server if a search engine tried to index them. For a more complete set of examples of `robots.txt`, refer to The Web Robots Page.

Testing Access Permissions

Using the data presented earlier, a simple crawler can test whether it is allowed to download a page using `RobotFileParser.can_fetch()`.

```
# urllib_robotparser_simple.py

from urllib import parse
from urllib import robotparser

AGENT_NAME = 'PyMOTW'
URL_BASE = 'https://pymotw.com/'
parser = robotparser.RobotFileParser()
parser.set_url(parse.urljoin(URL_BASE, 'robots.txt'))
parser.read()

PATHS = [
    '/',
    '/PyMOTW/',
    '/admin/',
    '/downloads/PyMOTW-1.92.tar.gz',
]

for path in PATHS:
    print('{!r:>6} : {}'.format(
        parser.can_fetch(AGENT_NAME, path), path))
    url = parse.urljoin(URL_BASE, path)
    print('{!r:>6} : {}'.format(
        parser.can_fetch(AGENT_NAME, url), url))
    print()
```

The URL argument to `can_fetch()` can be a path relative to the root of the site, or full URL.

```
$ python3 urllib_robotparser_simple.py

  True : /
  True : https://pymotw.com/

  True : /PyMOTW/
  True : https://pymotw.com/PyMOTW/

 False : /admin/
 False : https://pymotw.com/admin/

 False : /downloads/PyMOTW-1.92.tar.gz
 False : https://pymotw.com/downloads/PyMOTW-1.92.tar.gz
```

Long-lived Spiders

An application that takes a long time to process the resources it downloads or that is throttled to pause between downloads should check for new `robots.txt` files periodically based on the age of the content it has downloaded already. The age is not managed automatically, but there are convenience methods to make tracking it easier.

```python
# urllib_robotparser_longlived.py

from urllib import robotparser
import time

AGENT_NAME = 'PyMOTW'
parser = robotparser.RobotFileParser()
# Using the local copy
parser.set_url('file:robots.txt')
parser.read()
parser.modified()

PATHS = [
    '/',
    '/PyMOTW/',
    '/admin/',
    '/downloads/PyMOTW-1.92.tar.gz',
]

for path in PATHS:
    age = int(time.time() - parser.mtime())
    print('age:', age, end=' ')
    if age > 1:
        print('rereading robots.txt')
        parser.read()
        parser.modified()
    else:
        print()
    print('{!r:>6} : {}'.format(
        parser.can_fetch(AGENT_NAME, path), path))
    # Simulate a delay in processing
    time.sleep(1)
    print()
```

This extreme example downloads a new `robots.txt` file if the one it has is more than one second old.

```
$ python3 urllib_robotparser_longlived.py

age: 0
  True : /

age: 1
  True : /PyMOTW/

age: 2 rereading robots.txt
 False : /admin/

age: 1
 False : /downloads/PyMOTW-1.92.tar.gz
False : /downloads/PyMOTW-1.92.tar.gz
```

A nicer version of the long-lived application might request the modification time for the file before downloading the entire thing. On the other hand, `robots.txt` files are usually fairly small, so it is not that much more expensive to just retrieve the entire document again.

See also

- Standard library documentation for urllib.robotparser
- The Web Robots Page – Description of `robots.txt` format.

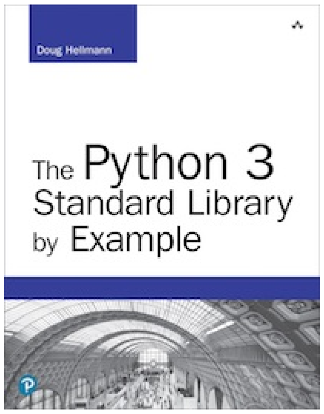
urllib.request — Network Resource Access

base64 — Encode Binary Data with ASCII

This page was last updated 2016-12-18.

Navigation

urllib.request — Network Resource Access
base64 — Encode Binary Data with ASCII



Get the book

The output from all the example programs from PyMOTW-3 has been generated with Python 3.7.1, unless otherwise noted. Some of the features described here may not be available in earlier versions of Python.

Looking for examples for Python 2?

This Site

Module Index
Index

© Copyright 2019, Doug Hellmann

Other Writing

Blog
The Python Standard Library By Example