
wsgi_lineprof

Release 0.18.0

Yusuke Miyazaki

Aug 27, 2023

CONTENTS

1	Contents	3
1.1	Quickstart	3
1.2	Examples	4
1.3	Configuration	6
1.4	Reference	8
1.5	Special Thanks	10
2	Links	13
3	Indices and tables	15
	Python Module Index	17
	Index	19

wsgi_lineprof is a WSGI middleware for line-by-line profiling.

wsgi_lineprof has the following features:

- *WSGI middleware*: It can be integrated with any WSGI-compatible applications and frameworks including Django, Pyramid, Flask, Bottle, and more.
- *Easily pluggable*: All configurations for profiling in one place. Users don't need to make any changes to their application.

wsgi_lineprof is *not recommended to be used in production environment* because of the overhead of profiling.

CONTENTS

1.1 Quickstart

1.1.1 Requirements

- CPython 3.8/3.9/3.10/3.11
- WSGI application

1.1.2 Installation

You can install `wsgi_lineprof` from PyPI.

```
$ pip install wsgi_lineprof
```

We provide wheel packages for some platforms. If your platform is supported, you don't need to build C extension. Otherwise, you need environment for building Python C extensions.

1.1.3 Enabling `wsgi_lineprof`

Apply `wsgi_lineprof` to the existing WSGI web application called `app`:

```
from wsgi_lineprof.middleware import LineProfilerMiddleware  
  
app = LineProfilerMiddleware(app)
```

1.1.4 Running `wsgi_lineprof`

Start the web application and access the application. `wsgi_lineprof` writes results to stdout every time an HTTP request is processed by default. You can see the output like this in your console:

```
... (snip) ...  
  
File: ./app.py  
Name: index  
Total time: 1.00518 [sec]  
  Line      Hits      Time  Code  
=====
```

(continues on next page)

(continued from previous page)

```
9             @app.route('/')
10             def index():
11                 1         1005175         time.sleep(1)
12                 1         4             return "Hello world!!"
... (snip) ...
```

Also, you can check the result on your web browser by accessing the special endpoint `/wsgi_lineprof/`.

1.1.5 Next Steps

- See [Examples](#) for more examples of integrating wsgi_lineprof with your applications.
- See [Configuration](#) for advanced usage such as filtering results, writing results to the file, accumulating results, and so on.

1.2 Examples

1.2.1 wsgiref

An example of using wsgi_lineprof with [wsgiref](#).

```
from wsgiref.simple_server import demo_app, make_server

from wsgi_lineprof.middleware import LineProfilerMiddleware

app = LineProfilerMiddleware(demo_app)

if __name__ == "__main__":
    with make_server('', 8000, app) as httpd:
        print("Serving HTTP on port 8000...")
        httpd.serve_forever()
```

1.2.2 Bottle

Examples of using wsgi_lineprof with [Bottle](#).

```
import bottle

from wsgi_lineprof.middleware import LineProfilerMiddleware

@bottle.route('/hello/<name>')
def index(name):
    return bottle.template('<b>Hello {{name}}</b>!', name=name)

app = LineProfilerMiddleware(bottle.app())
```

(continues on next page)

(continued from previous page)

```

if __name__ == "__main__":
    bottle.run(host='localhost', port=8080, app=app)

import bottle

from wsgi_lineprof.middleware import LineProfilerMiddleware

app = bottle.app()

@app.route('/hello/<name>')
def index(name):
    return bottle.template('<b>Hello {{name}}</b>!', name=name)

app = LineProfilerMiddleware(app)

if __name__ == "__main__":
    bottle.run(host='localhost', port=8080, app=app)

```

1.2.3 Flask

An example of using wsgi_lineprof with [Flask](#).

```

from flask import Flask
from wsgi_lineprof.middleware import LineProfilerMiddleware

app = Flask(__name__)

@app.route("/")
def hello():
    return "Hello, World!"

app.wsgi_app = LineProfilerMiddleware(app.wsgi_app)

if __name__ == '__main__':
    app.run(port=8000)

```

1.2.4 Django

An example of using wsgi_lineprof with [Django](#). We can load wsgi_lineprof in `<YOUR_PROJECT>.wsgi.py`.

```

import os

from django.core.wsgi import get_wsgi_application
from wsgi_lineprof.middleware import LineProfilerMiddleware

os.environ.setdefault('DJANGO_SETTINGS_MODULE', '<YOUR_PROJECT>.settings')

application = get_wsgi_application()
application = LineProfilerMiddleware(application)

```

1.3 Configuration

This page describes configuration options of wsgi_lineprof. You can provide various options as keyword arguments to change the behavior of the profiler:

```
from wsgi_lineprof.middleware import LineProfilerMiddleware
app = LineProfilerMiddleware(app, **options)
```

1.3.1 Filters

Users can get results from specific files or sort results by using filters. For example, use `FilenameFilter` to filter results with filename and use `TotalTimeSorter` to sort results by `total_time`.

```
import time

import bottle
from wsgi_lineprof.filters import FilenameFilter, TotalTimeSorter
from wsgi_lineprof.middleware import LineProfilerMiddleware

app = bottle.default_app()

def get_name():
    # Get some data...
    time.sleep(1)
    return "Monty Python"

@app.route('/')
def index():
    name = get_name()
    return "Hello, {}!!".format(name)

if __name__ == "__main__":
    filters = [
        # Results which filename contains "app2.py"
        FilenameFilter("app2.py"),
        # Sort by total time of results
        TotalTimeSorter(),
    ]
    # Add wsgi_lineprof as a WSGI middleware
    app = LineProfilerMiddleware(app, filters=filters)

    bottle.run(app=app)
```

Run the above script to start web server, then access <http://127.0.0.1:8080>. You can see results in stdout.

```
$ ./app2.py
Bottle v0.12.10 server starting up (using WSGIRefServer())...
Listening on http://127.0.0.1:8080/
Hit Ctrl-C to quit.

Time unit: 1e-06 [sec]
```

(continues on next page)

(continued from previous page)

```
File: ./app2.py
Name: index
Total time: 1.00526 [sec]
  Line      Hits      Time  Code
=====
   15                      @app.route('/')
   16                      def index():
   17      1      1005250      name = get_name()
   18      1          11      return "Hello, {}!".format(name)
```

```
File: ./app2.py
Name: get_name
Total time: 1.00523 [sec]
  Line      Hits      Time  Code
=====
   10                      def get_name():
   11                      # Get some data...
   12      1      1005226      time.sleep(1)
   13      1          4      return "Monty Python"

127.0.0.1 - - [30/Nov/2016 17:21:12] "GET / HTTP/1.1" 200 21
```

There are more useful filters in `wsgi_lineprof.filters`. Examples:

- `FilenameFilter("(file1|file2).py", regex=True)`
- `NameFilter("(fun1|fun2).py", regex=True)`

1.3.2 Stream

By using `stream` option, you can output results to a file. For example, you can output logs to `lineprof.log`.

```
f = open("lineprof.log", "w")
app = LineProfilerMiddleware(app, stream=f)
bottle.run(app=app)
```

1.3.3 Async Stream

By using `async_stream` option, `wsgi_lineprof` starts a new thread for writing results. This option is useful when you do not want the main thread blocked for writing results.

```
# Start a new thread for writing results
app = LineProfilerMiddleware(app, async_stream=True)
bottle.run(app=app)
```

1.3.4 Accumulate Mode

By default, wsgi_lineprof writes results every time a request is processed. By enabling accumulate option, wsgi_lineprof accumulate results of all requests and writes the result on interpreter termination.

```
app = LineProfilerMiddleware(app, accumulate=True)
bottle.run(app=app)
```

1.3.5 Colorize Output

Colorized output is enabled by default for stdout and stderr. You can disable the feature using the color option.

```
app = LineProfilerMiddleware(app, color=False)
bottle.run(app=app)
```

1.3.6 Result Endpoint

By default, you can access an endpoint /wsgi_lineprof/ to see the results. This endpoint is configurable.

```
app = LineProfilerMiddleware(app, endpoint='/custom_result_endpoint/')
bottle.run(app=app)
```

1.4 Reference

1.4.1 wsgi_lineprof.filters module

class wsgi_lineprof.filters.BaseFilter

Bases: object

abstract filter(stats: Iterable[LineProfilerStat]) → Iterable[LineProfilerStat]

class wsgi_lineprof.filters.FilenameFilter(filename: str, regex: bool = False)

Bases: BaseFilter

Filter which matches with filename

filter(stats: Iterable[LineProfilerStat]) → Iterable[LineProfilerStat]

class wsgi_lineprof.filters.NameFilter(name: str, regex: bool = True)

Bases: BaseFilter

Filter which matches with name

filter(stats: Iterable[LineProfilerStat]) → Iterable[LineProfilerStat]

class wsgi_lineprof.filters.TopItemsFilter(n: int = 10)

Bases: BaseFilter

Get first n stats

filter(stats: Iterable[LineProfilerStat]) → Iterable[LineProfilerStat]

```

class wsgi_lineprof.filters.TotalTimeSorter(reverse: bool = True)
    Bases: BaseFilter
    Sort stats by total time
    filter(stats: Iterable[LineProfilerStat]) → Iterable[LineProfilerStat]

```

1.4.2 wsgi_lineprof.middleware module

```

class wsgi_lineprof.middleware.LineProfilerMiddleware(app: WSGIApplication, stream:
    ~typing.TextIO | None = None, filters: ~typing.Iterable[~typing.Callable[~typing.Iterable[~wsgi_lineprof.s
    ~typ-
    ing.Iterable[~wsgi_lineprof.stats.LineProfilerStat]]
    | ~wsgi_lineprof.filters.BaseFilter] = (),
    async_stream: bool = False, accumulate:
    bool = False, color: bool = True,
    profiler_class: ~typ-
    ing.Type[~wsgi_lineprof.profiler.LineProfiler]
    = <class
    'wsgi_lineprof.profiler.LineProfiler'>,
    endpoint: str = '/wsgi_lineprof/')

    Bases: object

```

1.4.3 wsgi_lineprof.profiler module

```

class wsgi_lineprof.profiler.LineProfiler
    Bases: LineProfiler

```

1.4.4 wsgi_lineprof.stats module

```

class wsgi_lineprof.stats.LineProfilerStat(code: code, timings: Dict[int, LineTiming])
    Bases: object
    property filename: str
    property firstlineno: int
    property name: str

class wsgi_lineprof.stats.LineProfilerStats(stats: Iterable[LineProfilerStat], unit: float)
    Bases: object
    filter(f: Callable[[Iterable[LineProfilerStat]], Iterable[LineProfilerStat]] | BaseFilter) → LineProfilerStats
    classmethod from_measurement_and_unit(measurement: Dict[code, Dict[int, LineTiming]], unit: float)
        → LineProfilerStats
    classmethod from_request_measurement(request_measurement: RequestMeasurement) →
        LineProfilerStats

```

1.4.5 wsgi_lineprof.types module

```
class wsgi_lineprof.types.RequestMeasurement
    Bases: TypedDict
    elapsed: float
    id: UUID
    path_info: str
    query_string: str
    request_method: str
    results: Dict[code, Dict[int, LineTiming]]
    started_at: datetime
    unit: float
```

1.4.6 wsgi_lineprof.writers module

```
class wsgi_lineprof.writers.AsyncStreamWriter(stream: TextIO, formatter: BaseFormatter)
    Bases: BaseStreamWriter
    write(stats: LineProfilerStats) → None

class wsgi_lineprof.writers.BaseStreamWriter(stream: TextIO, formatter: BaseFormatter, *kwargs:
    Any)
    Bases: object
    abstract write(stats: LineProfilerStats) → None

class wsgi_lineprof.writers.SyncStreamWriter(stream: TextIO, formatter: BaseFormatter, *kwargs:
    Any)
    Bases: BaseStreamWriter
    write(stats: LineProfilerStats) → None
```

1.4.7 Module contents

1.5 Special Thanks

This project uses code from the following project:

- [rkern/line_profiler](#)

This project is inspired by the following project:

- [kainosnoema/rack-lineprof](#)

wsgi_lineprof is integrated with the following projects:

- [kobinpy/wsgicli](#)
- [denzow/wsgi_lineprof_reporter](#)

wsgi_lineprof is mentioned in the following entries:

- [1 Peter WangPythonWebPyCon JP 2017gihyo.jp ...](#)
- [DjangoDjangoWSGI -](#)
- [PythonWSGI\(wsgi_lineprof\) - \[Dd\]enzow\(ill\)? with DB and Python](#)
- [PythonWSGI\(wsgi_lineprof_reporter\) - \[Dd\]enzow\(ill\)? with DB and Python](#)
- [Server-side development — c2cgeoportal documentation](#)

LINKS

- [GitHub: ymyzk/wsgi_lineprof](#)
- [WSGI – ymyzk’s blog](#)
- [Python wsgi_lineprof – ymyzk’s blog](#)
- [Python // Implementation of a profiler for Python web applications - PyCon JP 2019](#)

INDICES AND TABLES

- `genindex`
- `modindex`
- `search`

PYTHON MODULE INDEX

W

- `wsgi_lineprof`, 10
- `wsgi_lineprof.filters`, 8
- `wsgi_lineprof.middleware`, 9
- `wsgi_lineprof.profiler`, 9
- `wsgi_lineprof.stats`, 9
- `wsgi_lineprof.types`, 10
- `wsgi_lineprof.writers`, 10

INDEX

A

`AsyncStreamWriter` (class in `wsgi_lineprof.writers`), 10

B

`BaseFilter` (class in `wsgi_lineprof.filters`), 8

`BaseStreamWriter` (class in `wsgi_lineprof.writers`), 10

E

`elapsed` (`wsgi_lineprof.types.RequestMeasurement` attribute), 10

F

`filename` (`wsgi_lineprof.stats.LineProfilerStat` property), 9

`FilenameFilter` (class in `wsgi_lineprof.filters`), 8

`filter()` (`wsgi_lineprof.filters.BaseFilter` method), 8

`filter()` (`wsgi_lineprof.filters.FilenameFilter` method), 8

`filter()` (`wsgi_lineprof.filters.NameFilter` method), 8

`filter()` (`wsgi_lineprof.filters.TopItemsFilter` method), 8

`filter()` (`wsgi_lineprof.filters.TotalTimeSorter` method), 9

`filter()` (`wsgi_lineprof.stats.LineProfilerStats` method), 9

`firstlineno` (`wsgi_lineprof.stats.LineProfilerStat` property), 9

`from_measurement_and_unit()`
(`wsgi_lineprof.stats.LineProfilerStats` class method), 9

`from_request_measurement()`
(`wsgi_lineprof.stats.LineProfilerStats` class method), 9

I

`id` (`wsgi_lineprof.types.RequestMeasurement` attribute), 10

L

`LineProfiler` (class in `wsgi_lineprof.profiler`), 9

`LineProfilerMiddleware` (class in `wsgi_lineprof.middleware`), 9

`LineProfilerStat` (class in `wsgi_lineprof.stats`), 9

`LineProfilerStats` (class in `wsgi_lineprof.stats`), 9

M

module

`wsgi_lineprof`, 10

`wsgi_lineprof.filters`, 8

`wsgi_lineprof.middleware`, 9

`wsgi_lineprof.profiler`, 9

`wsgi_lineprof.stats`, 9

`wsgi_lineprof.types`, 10

`wsgi_lineprof.writers`, 10

N

`name` (`wsgi_lineprof.stats.LineProfilerStat` property), 9

`NameFilter` (class in `wsgi_lineprof.filters`), 8

P

`path_info` (`wsgi_lineprof.types.RequestMeasurement` attribute), 10

Q

`query_string` (`wsgi_lineprof.types.RequestMeasurement` attribute), 10

R

`request_method` (`wsgi_lineprof.types.RequestMeasurement` attribute), 10

`RequestMeasurement` (class in `wsgi_lineprof.types`), 10

`results` (`wsgi_lineprof.types.RequestMeasurement` attribute), 10

S

`started_at` (`wsgi_lineprof.types.RequestMeasurement` attribute), 10

`SyncStreamWriter` (class in `wsgi_lineprof.writers`), 10

T

`TopItemsFilter` (class in `wsgi_lineprof.filters`), 8

`TotalTimeSorter` (class in `wsgi_lineprof.filters`), 8

U

unit (wsgi_lineprof.types.RequestMeasurement attribute), 10

W

write() (wsgi_lineprof.writers.AsyncStreamWriter method), 10

write() (wsgi_lineprof.writers.BaseStreamWriter method), 10

write() (wsgi_lineprof.writers.SyncStreamWriter method), 10

wsgi_lineprof
module, 10

wsgi_lineprof.filters
module, 8

wsgi_lineprof.middleware
module, 9

wsgi_lineprof.profiler
module, 9

wsgi_lineprof.stats
module, 9

wsgi_lineprof.types
module, 10

wsgi_lineprof.writers
module, 10