Who I Am

- Daniel Leech
- From England
- Open Source Developer
- Bicycle Tourer

Open Source Developer

- Symfony CMF Core Team
- Sulu CMS
- PHPCR-ODM
- CMF Routing Auto
- PHPCR Shell
- Jackalope FS

Bicycle Tour 2015

Austria

Slovakia

Hungary

Romania

Bulgaria

Turkey, Istanbul!

Why Benchmarking?

Jackalope 2

(this is not the real logo)

PHPCR

Benchmarking

What is Benchmarking?

“In computing, a benchmark is the act of running a computer program, a set of programs, or other operations, in order to assess the ***relative*** performance of an object, normally by running a number of standard tests and trials against it”

- Wikipedia

What I Talk About When I Talk About Benchmarking

“The measurement of time taken to execute some PHP code compared to the same measurement of equivalent PHP code”

- Me

Its All About Microtime!

```
$start = microtime(true);
// do something
$timeTaken = microtime(true) - $start;
echo 'Time 1: ' . $timeTaken;

$start = microtime(true);
// do something else
$timeTaken = microtime(true) - $start;
echo 'Time 2: ' . $timeTaken;
```

About Microtime

- Amount of time elapsed since the UNIX epoch acurate to a microsecond.
- It uses the symbol µs (Greek “mu”)
- 1 microsecond = 1,000,000th (10^6) of a second!
- Highest degree of precision available in PHP.

Microseconds

- Human eye blink takes 350,000 microseconds (just over 1/3 of one second).
- Human finger click takes 150,000 microseconds (just over 1/7 of one second).
- Some PHP functions execute < 1µs

Why Would You Want to Benchmark?

- Optimize algorithms.
- Prevent performance regressions in new code.
- Compare relative performance of interchangable dependencies.
- Catch memory leaks.
- Empirical knowledge.

Compare

```
isset($array['foo'])
```

Functions

```
array_key_exists('foo', '$array)
```

Compare Branches

master

VS

harder_better_stronger_faster

FIGHT

Compare

MySQL™

VS

PostgreSQL

FIGHT

Implementations

Memory Leaks

Empirical Knowledge

Conclusions

- We benchmark using microtime, because thats the best we have.
- Benchmarking can help you make smart decisions.

Questions?

PHPBench

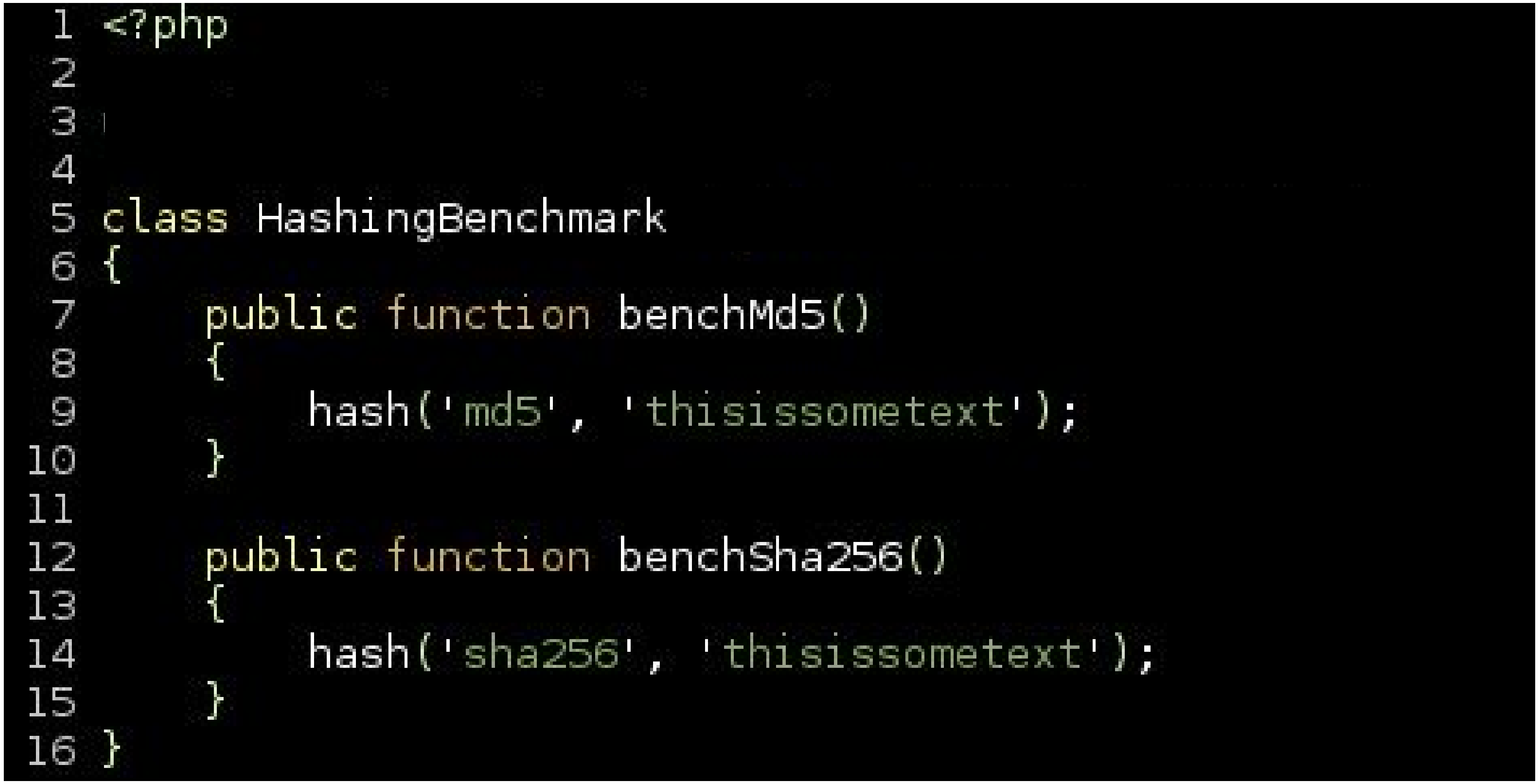
“PHPBench is a Benchmark Runner for PHP which can Generate Reports”

- J. Lennon

Benchmark Classes

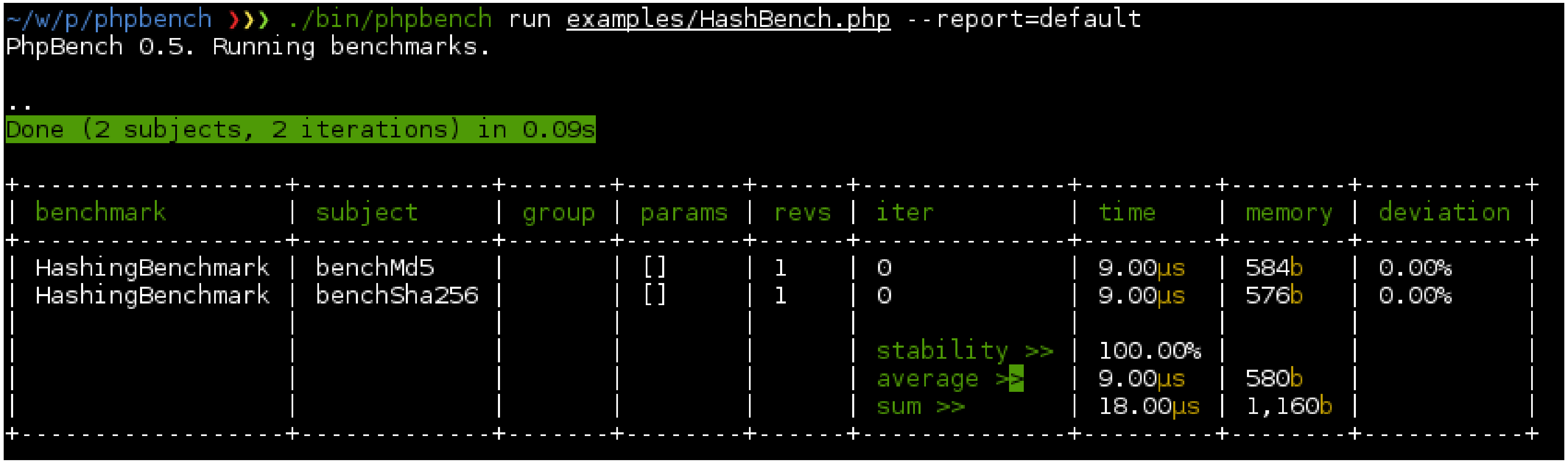
- Must be suffixed with “Bench”.
- Similar to PHPUnit
- Have their own autoloading env.
- Do not depend on the PHPBench library (not abstract classes, no interfaces).

Benchmarking Hash Algorithms



```php
<?php



class HashingBenchmark
{
    public function benchMd5()
    {
        hash('md5', 'thisissometext');
    }

    public function benchSha256()
    {
        hash('sha256', 'thisissometext');
    }
}
```

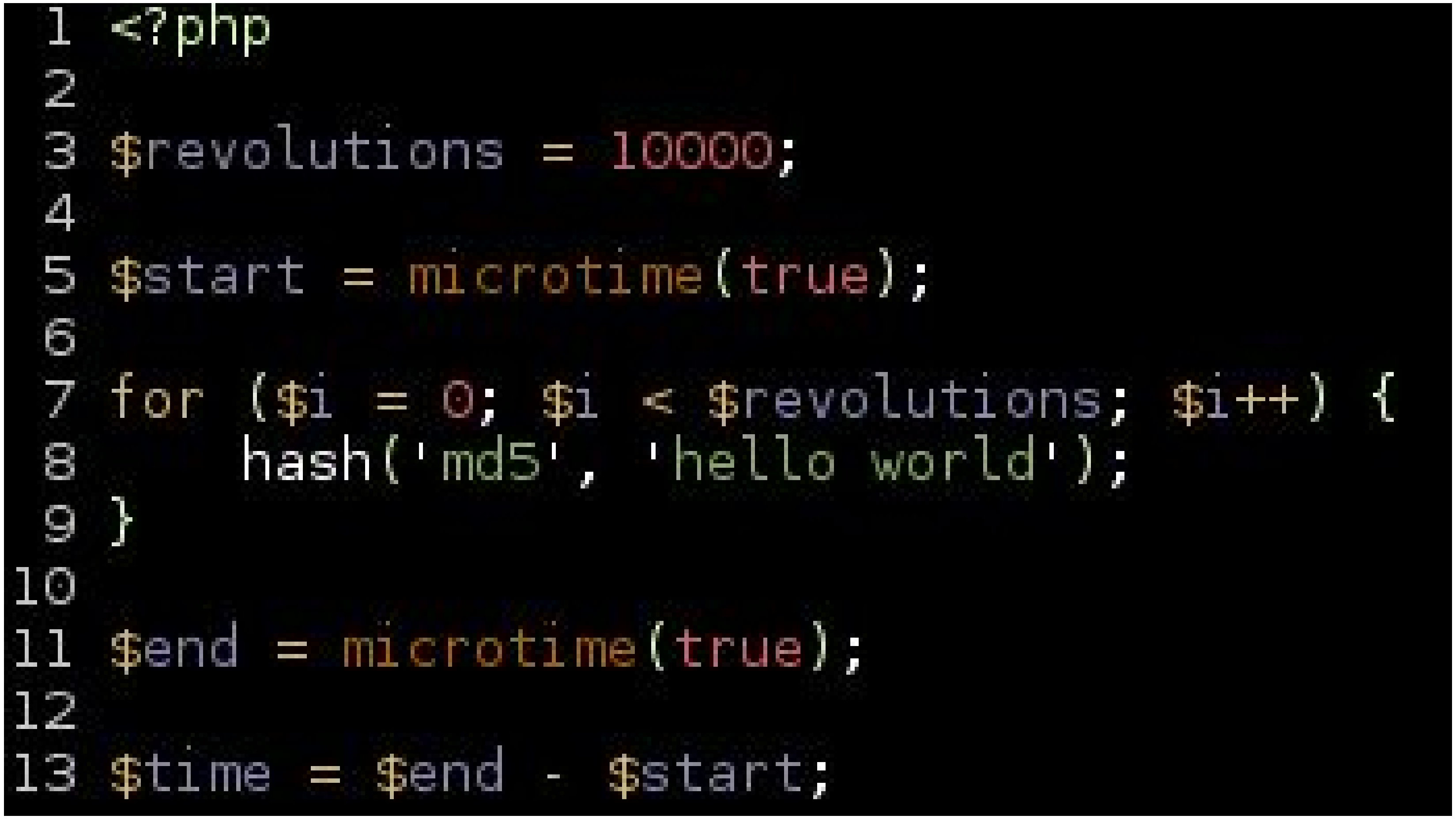
Benchmarking Hash Algorithms



```
~/w/p/phpbench ❯❯❯ ./bin/phpbench run examples/HashBench.php --report=default
PhpBench 0.5. Running benchmarks.

..
Done (2 subjects, 2 iterations) in 0.09s

+------------------+-------------+-------+--------+------+--------------+---------+--------+-----------+
| benchmark        | subject     | group | params | revs | iter         | time    | memory | deviation |
+------------------+-------------+-------+--------+------+--------------+---------+--------+-----------+
| HashingBenchmark | benchMd5    |       | []     | 1    | 0            | 9.00µs  | 584b   | 0.00%     |
| HashingBenchmark | benchSha256 |       | []     | 1    | 0            | 9.00µs  | 576b   | 0.00%     |
|                  |             |       |        |      |              |         |        |           |
|                  |             |       |        |      | stability >> | 100.00% |        |           |
|                  |             |       |        |      | average >>   | 9.00µs  | 580b   |           |
|                  |             |       |        |      | sum >>       | 18.00µs | 1,160b |           |
+------------------+-------------+-------+--------+------+--------------+---------+--------+-----------+
```

Revolutions

Revolutions



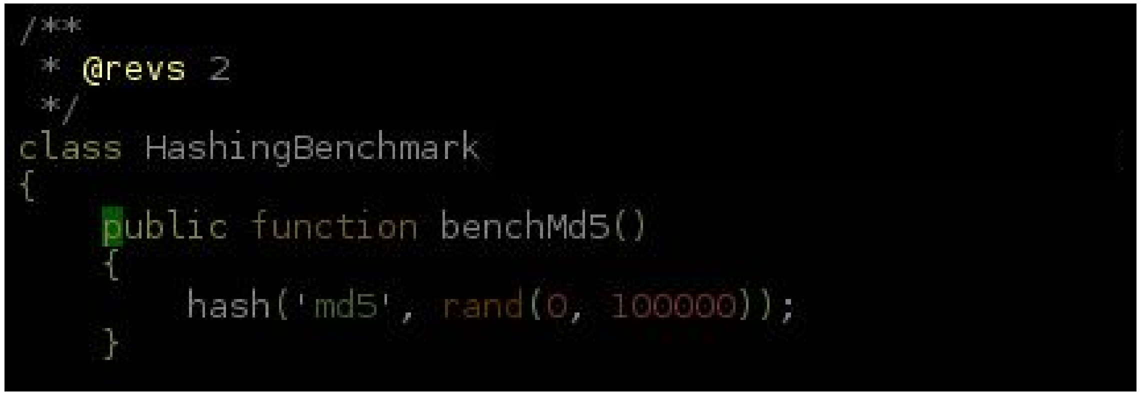
```php
<?php

$revolutions = 10000;

$start = microtime(true);

for ($i = 0; $i < $revolutions; $i++) {
    hash('md5', 'hello world');
}

$end = microtime(true);

$time = $end - $start;
```

Revolutions Increase Precision

```
time = total time / revolutions
```

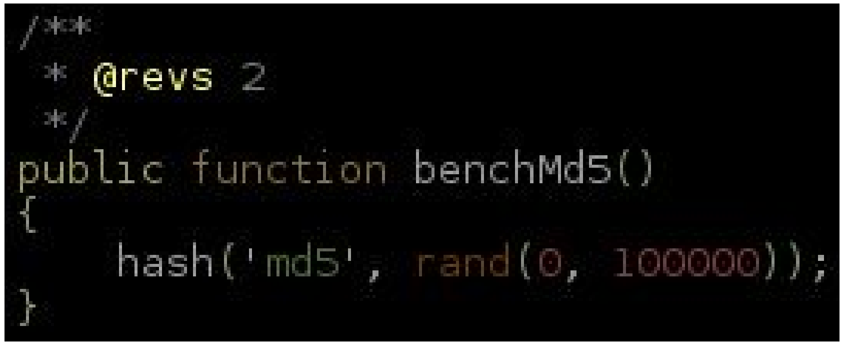
Revolutions	MD5 Time	SHA256 Time
1	9ms	9ms
10	4.6ms	5.00ms
100	2.7ms	3.09ms
1000	1.87ms	2.20ms
10000	1.62ms	2.25ms

Specifying Revolutions



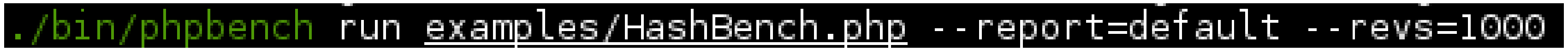
```
/**
 * @revs 2
 */
class HashingBenchmark
{
    public function benchMd5()
    {
        hash('md5', rand(0, 100000));
    }
```

Default revolutions for class



```
/**
 * @revs 2
 */
public function benchMd5()
{
    hash('md5', rand(0, 100000));
}
```

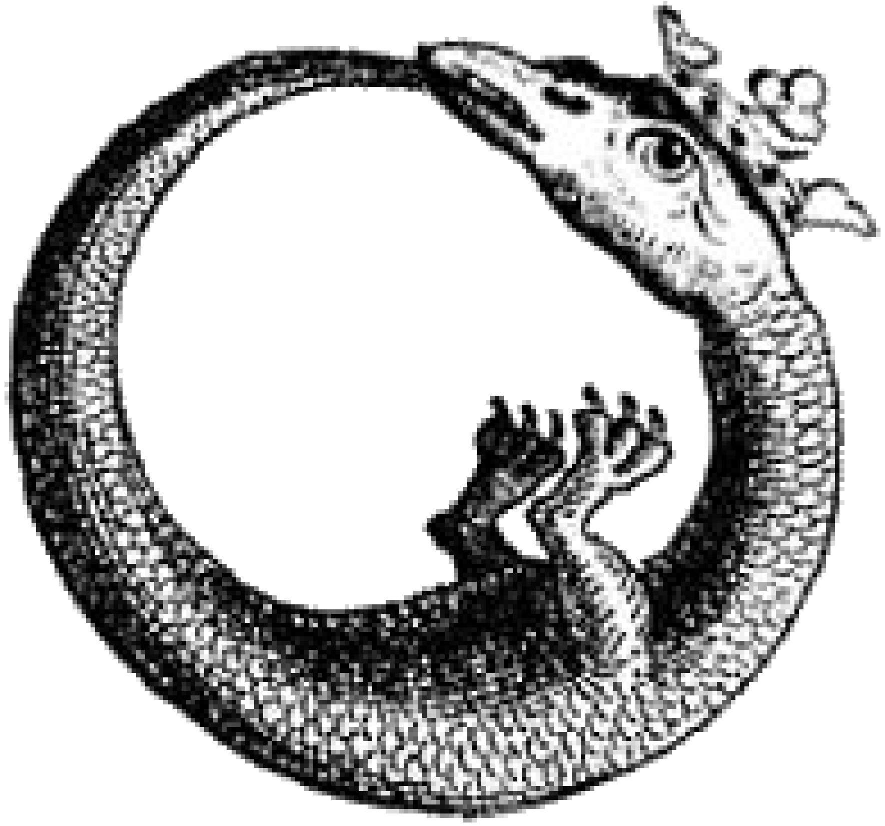
Revolutions per method



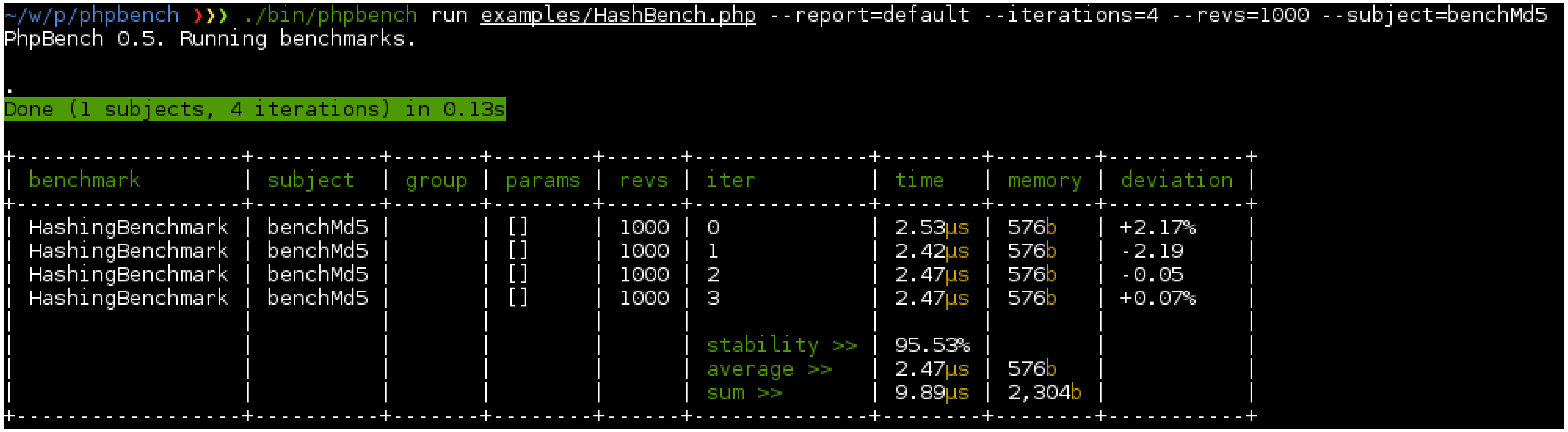
```
./bin/phpbench run examples/HashBench.php --report=default --revs=1000
```

Revolutions overridden on command line

Iterations



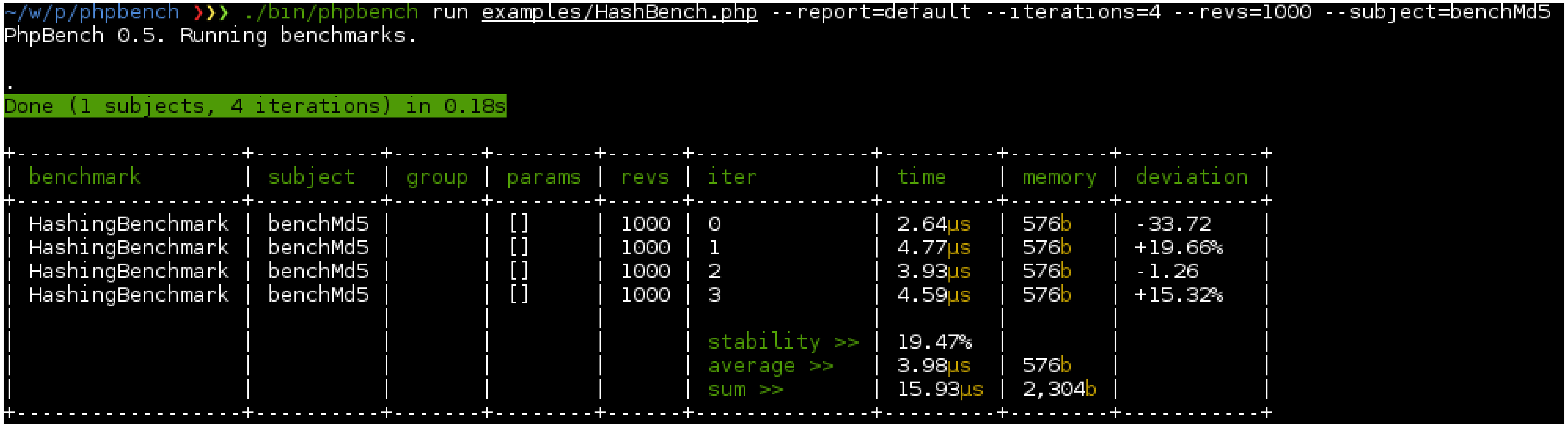
Iterations Confirm Stability


```
~/w/p/phpbench ❯❯❯ ./bin/phpbench run examples/HashBench.php --report=default --iterations=4 --revs=1000 --subject=benchMd5
PhpBench 0.5. Running benchmarks.

.
Done (1 subjects, 4 iterations) in 0.13s

+------------------+----------+-------+--------+------+--------------+--------+--------+-----------+
| benchmark        | subject  | group | params | revs | iter         | time   | memory | deviation |
+------------------+----------+-------+--------+------+--------------+--------+--------+-----------+
| HashingBenchmark | benchMd5 |       | []     | 1000 | 0            | 2.53μs | 576b   | +2.17%    |
| HashingBenchmark | benchMd5 |       | []     | 1000 | 1            | 2.42μs | 576b   | -2.19     |
| HashingBenchmark | benchMd5 |       | []     | 1000 | 2            | 2.47μs | 576b   | -0.05     |
| HashingBenchmark | benchMd5 |       | []     | 1000 | 3            | 2.47μs | 576b   | +0.07%    |
|                  |          |       |        |      |              |        |        |           |
|                  |          |       |        |      | stability >> | 95.53% |        |           |
|                  |          |       |        |      | average >>   | 2.47μs | 576b   |           |
|                  |          |       |        |      | sum >>       | 9.89μs | 2,304b |           |
+------------------+----------+-------+--------+------+--------------+--------+--------+-----------+
```

good stability

Stability 95.52%

Iterations Confirm Stability

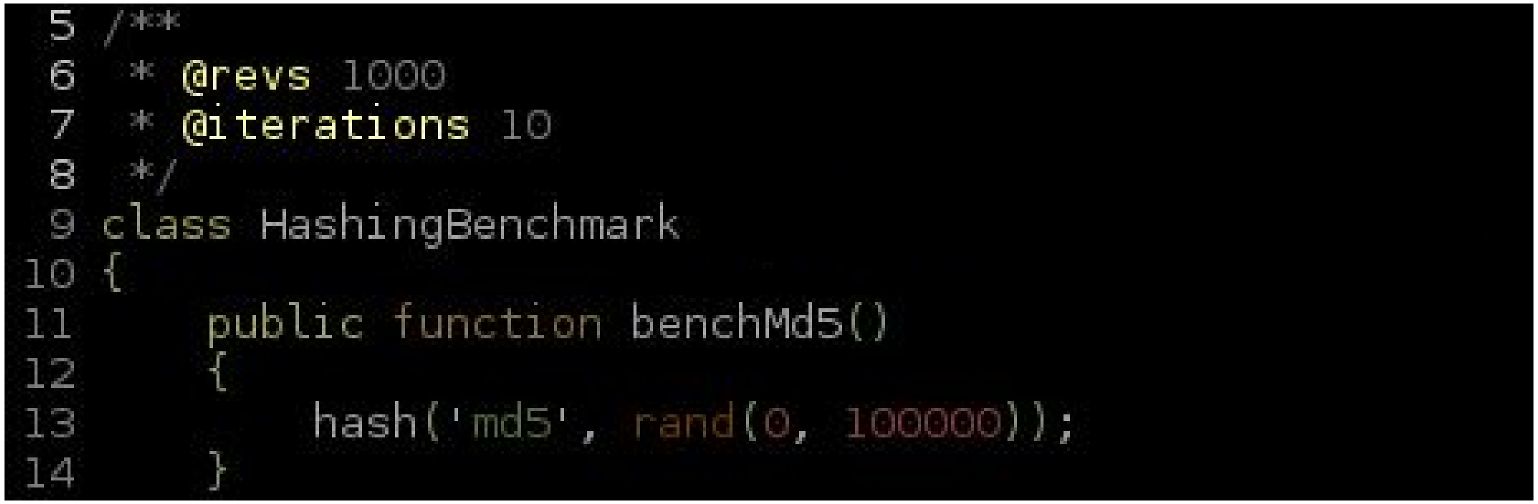


```
~/w/p/phpbench ❯❯❯ ./bin/phpbench run examples/HashBench.php --report=default --iterations=4 --revs=1000 --subject=benchMd5
PhpBench 0.5. Running benchmarks.

.
Done (1 subjects, 4 iterations) in 0.18s

+------------------+---------+--------+--------+------+---------------+----------+--------+-----------+
| benchmark        | subject | group  | params | revs | iter          | time     | memory | deviation |
+------------------+---------+--------+--------+------+---------------+----------+--------+-----------+
| HashingBenchmark | benchMd5 |       | []     | 1000 | 0             | 2.64µs   | 576b   | -33.72    |
| HashingBenchmark | benchMd5 |       | []     | 1000 | 1             | 4.77µs   | 576b   | +19.66%   |
| HashingBenchmark | benchMd5 |       | []     | 1000 | 2             | 3.93µs   | 576b   | -1.26     |
| HashingBenchmark | benchMd5 |       | []     | 1000 | 3             | 4.59µs   | 576b   | +15.32%   |
|                  |         |        |        |      |               |          |        |           |
|                  |         |        |        |      | stability >>  | 19.47%   |        |           |
|                  |         |        |        |      | average >>    | 3.98µs   | 576b   |           |
|                  |         |        |        |      | sum >>        | 15.93µs  | 2,304b |           |
+------------------+---------+--------+--------+------+---------------+----------+--------+-----------+
```

bad stability

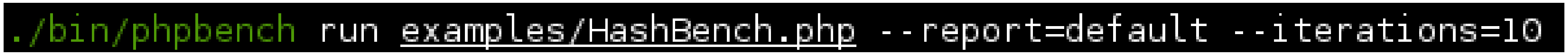
Stability 19.47%

Specifying Iterations



```
/**
 * @revs 1000
 * @iterations 10
 */
class HashingBenchmark
{
    public function benchMd5()
    {
        hash('md5', rand(0, 100000));
    }
```

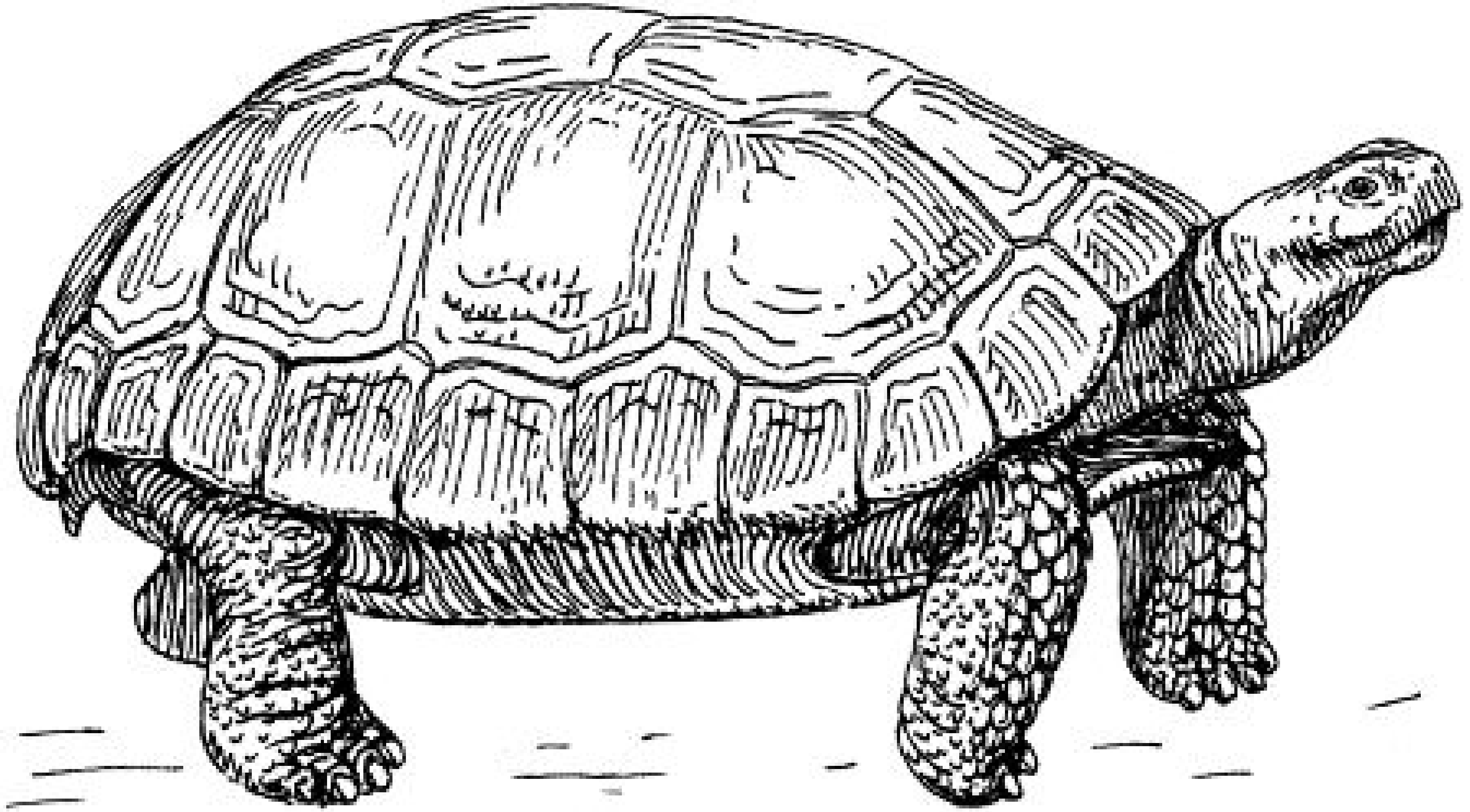
as a class or method annotation



```
./bin/phpbench run examples/HashBench.php --report=default --iterations=10
```

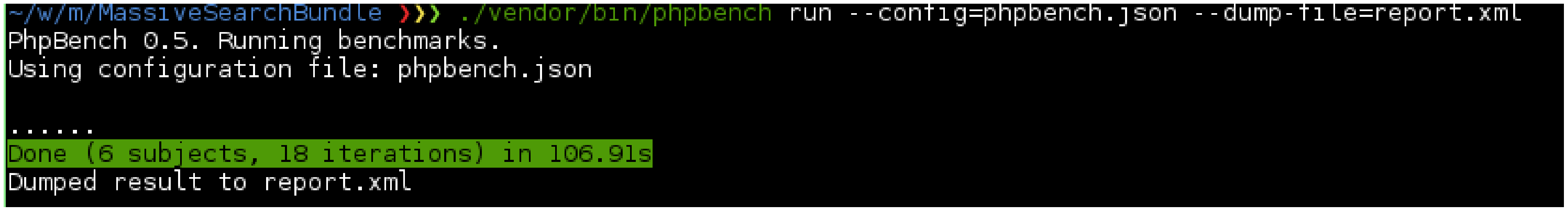
as a command line override

Benchmarks Can Be Slow



Benchmarking Slow Things

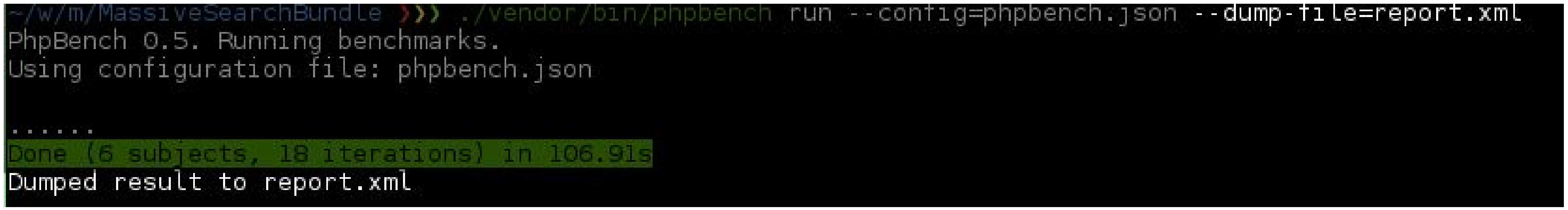
Indexing a large number of documents
in different search implementations



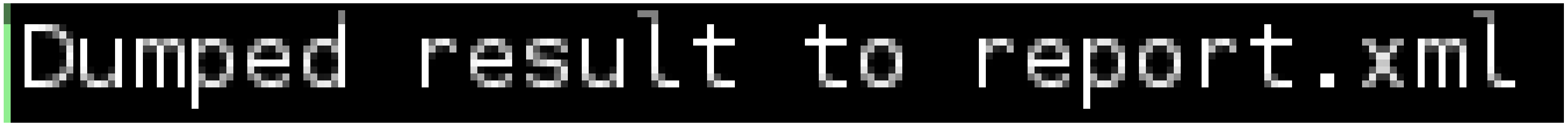
```
~/w/m/MassiveSearchBundle ❯❯❯ ./vendor/bin/phpbench run --config=phpbench.json --dump-file=report.xml
PhpBench 0.5. Running benchmarks.
Using configuration file: phpbench.json

......
Done (6 subjects, 18 iterations) in 106.91s
Dumped result to report.xml
```

1 minute 46 seconds!

Dumping the Results to XML

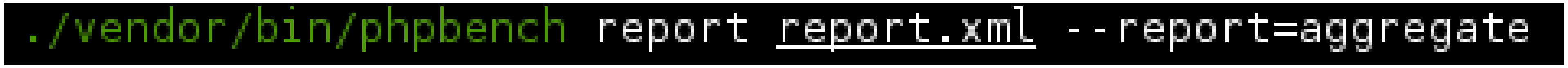


```
~/w/m/MassiveSearchBundle ❯❯❯ ./vendor/bin/phpbench run --config=phpbench.json --dump-file=report.xml
PhpBench 0.5. Running benchmarks.
Using configuration file: phpbench.json

......
Done (6 subjects, 18 iterations) in 106.91s
Dumped result to report.xml
```

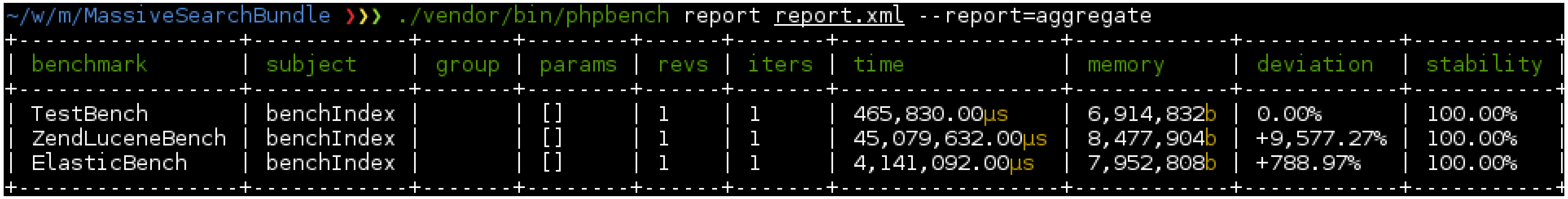


```
Dumped result to report.xml
```

Reporting on the XML file



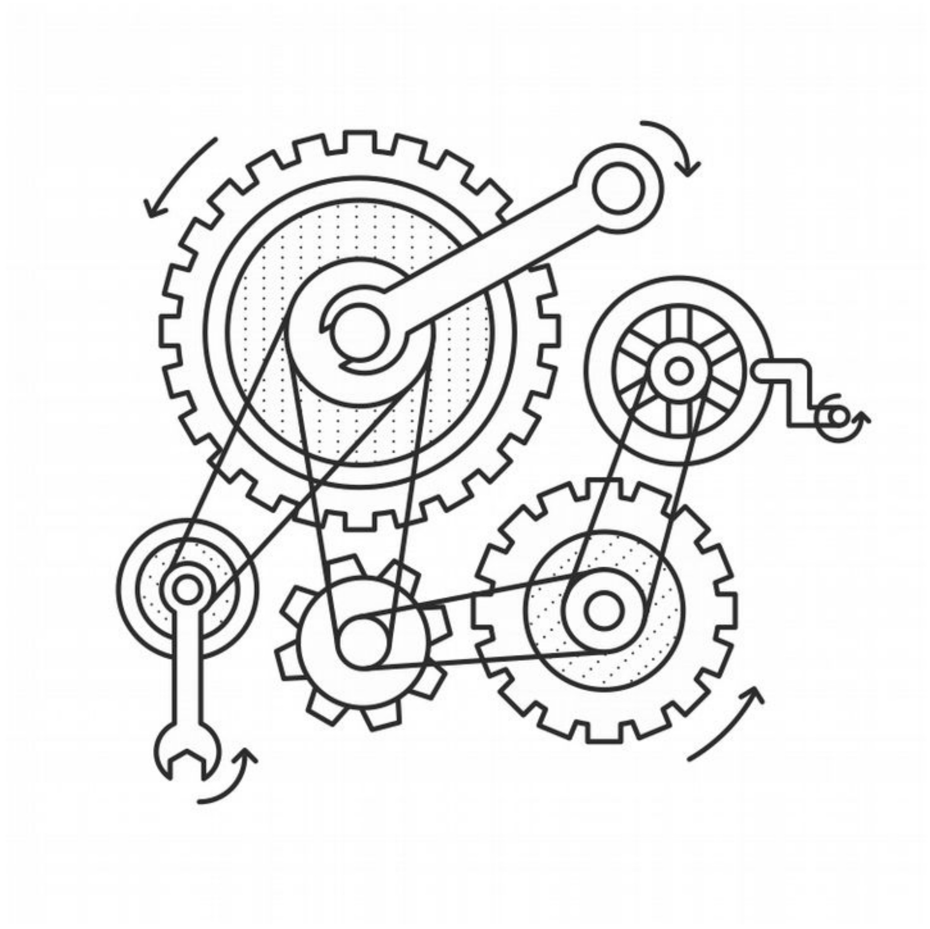
```
./vendor/bin/phpbench report report.xml --report=aggregate
```



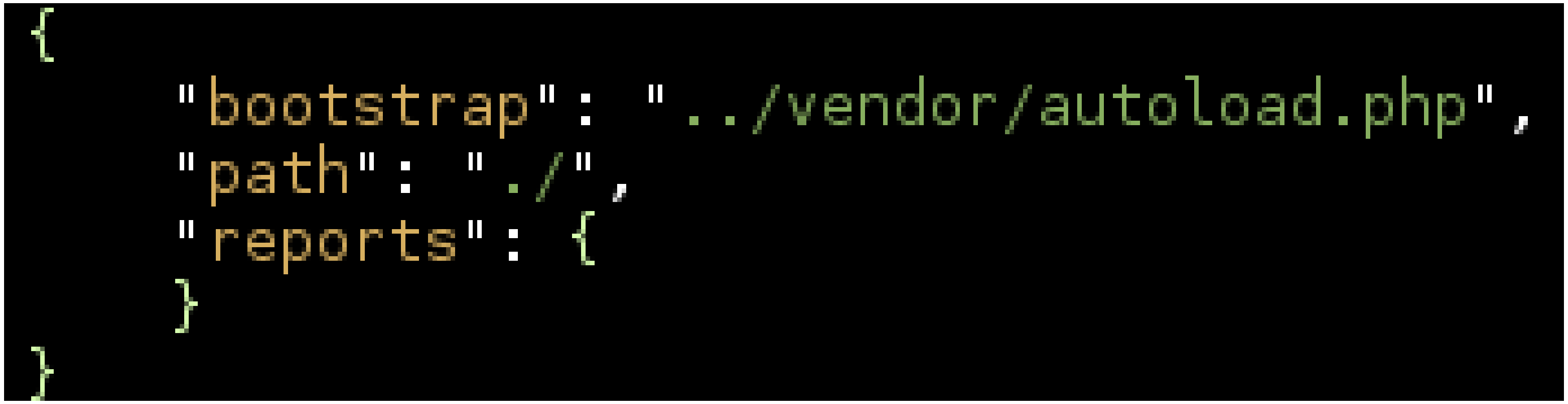
```
~/w/m/MassiveSearchBundle ❯❯❯ ./vendor/bin/phpbench report report.xml --report=aggregate
+-----------------+------------+-------+--------+------+-------+-----------------+------------+------------+-----------+
| benchmark       | subject    | group | params | revs | iters | time            | memory     | deviation  | stability |
+-----------------+------------+-------+--------+------+-------+-----------------+------------+------------+-----------+
| TestBench       | benchIndex |       | []     | 1    | 1     | 465,830.00µs    | 6,914,832b | 0.00%      | 100.00%   |
| ZendLuceneBench | benchIndex |       | []     | 1    | 1     | 45,079,632.00µs | 8,477,904b | +9,577.27% | 100.00%   |
| ElasticBench    | benchIndex |       | []     | 1    | 1     | 4,141,092.00µs  | 7,952,808b | +788.97%   | 100.00%   |
+-----------------+------------+-------+--------+------+-------+-----------------+------------+------------+-----------+
```

Instant Report!

Configuration



JSON Configuration File

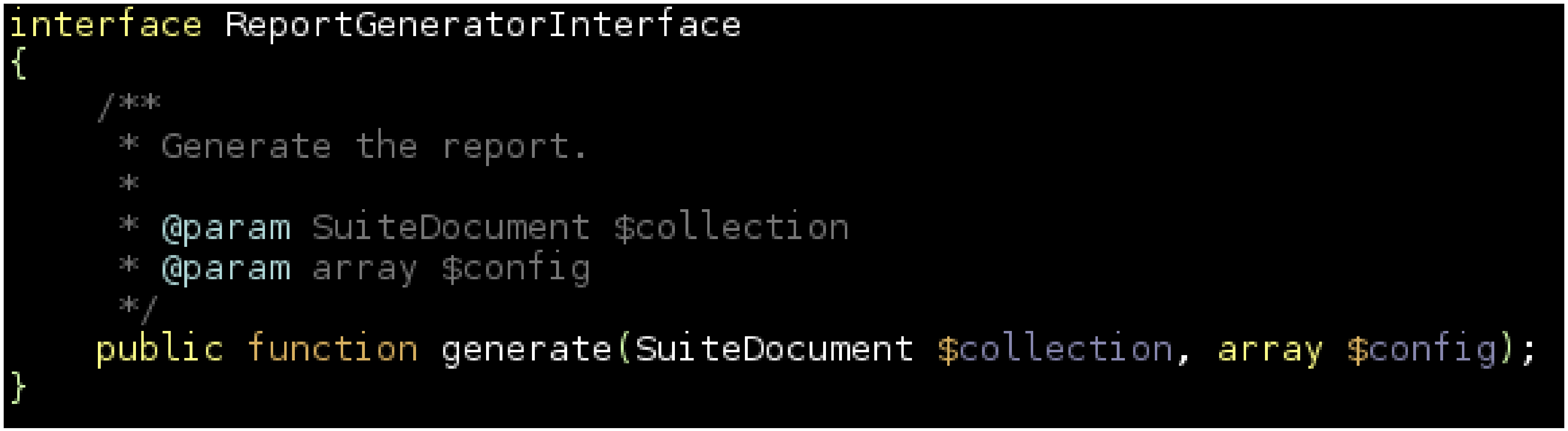


```
{
    "bootstrap": "../vendor/autoload.php",
    "path": "./",
    "reports": {
    }
}
```

- Automatically uses phpbench.json if present
- Config can be specified using --config

Reports

Report Generators



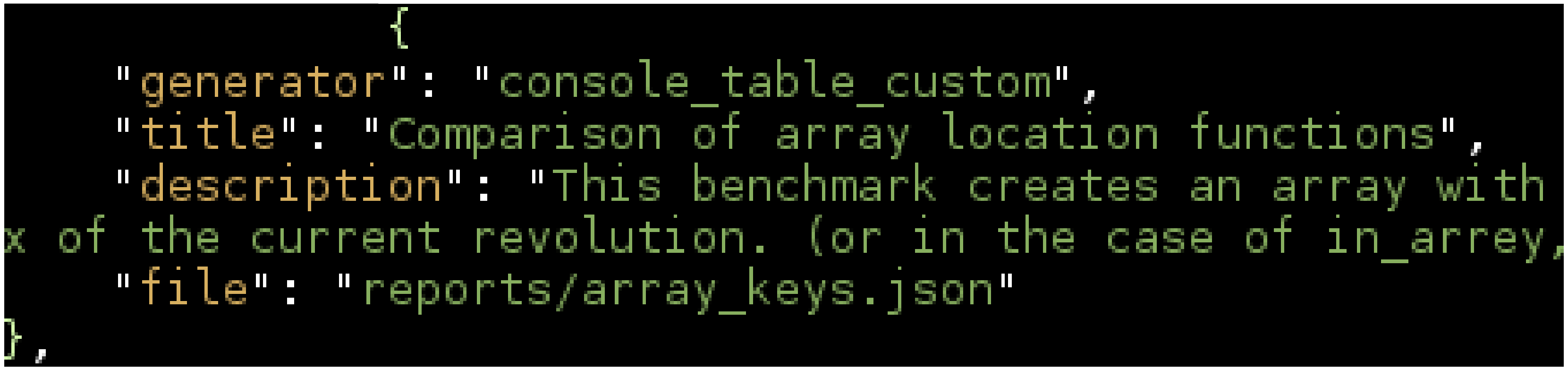
```
interface ReportGeneratorInterface
{
    /**
     * Generate the report.
     *
     * @param SuiteDocument $collection
     * @param array $config
     */
    public function generate(SuiteDocument $collection, array $config);
}
```

(simplified)

- ConsoleTableGenerator: Generates easy reports
- ConsoleTableCustomGenerator: Definition of complex reports
- CompositeGenerator: Generate multiple reports

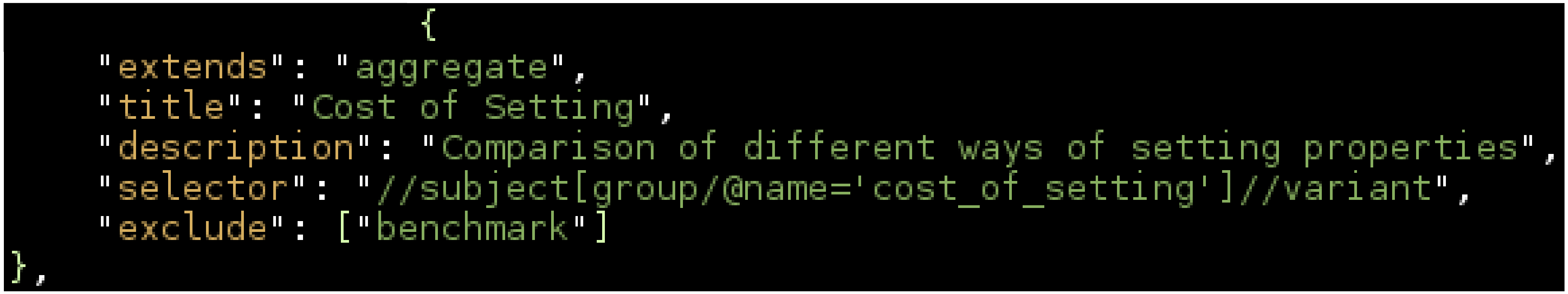
Reports

Reports are configured using an array



```
              {
    "generator": "console_table_custom",
    "title": "Comparison of array location functions",
    "description": "This benchmark creates an array with
x of the current revolution. (or in the case of in_arrey,
    "file": "reports/array_keys.json"
},
```

report using a specified generator

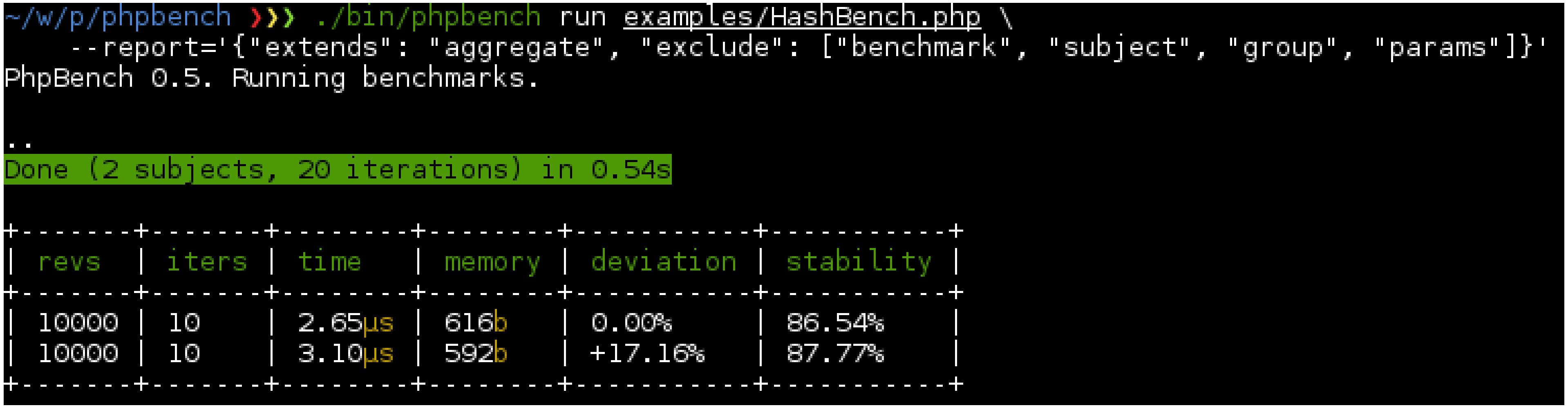


```
                  {
    "extends": "aggregate",
    "title": "Cost of Setting",
    "description": "Comparison of different ways of setting properties",
    "selector": "//subject[group/@name='cost_of_setting']//variant",
    "exclude": ["benchmark"]
},
```

report extending another report

Reports

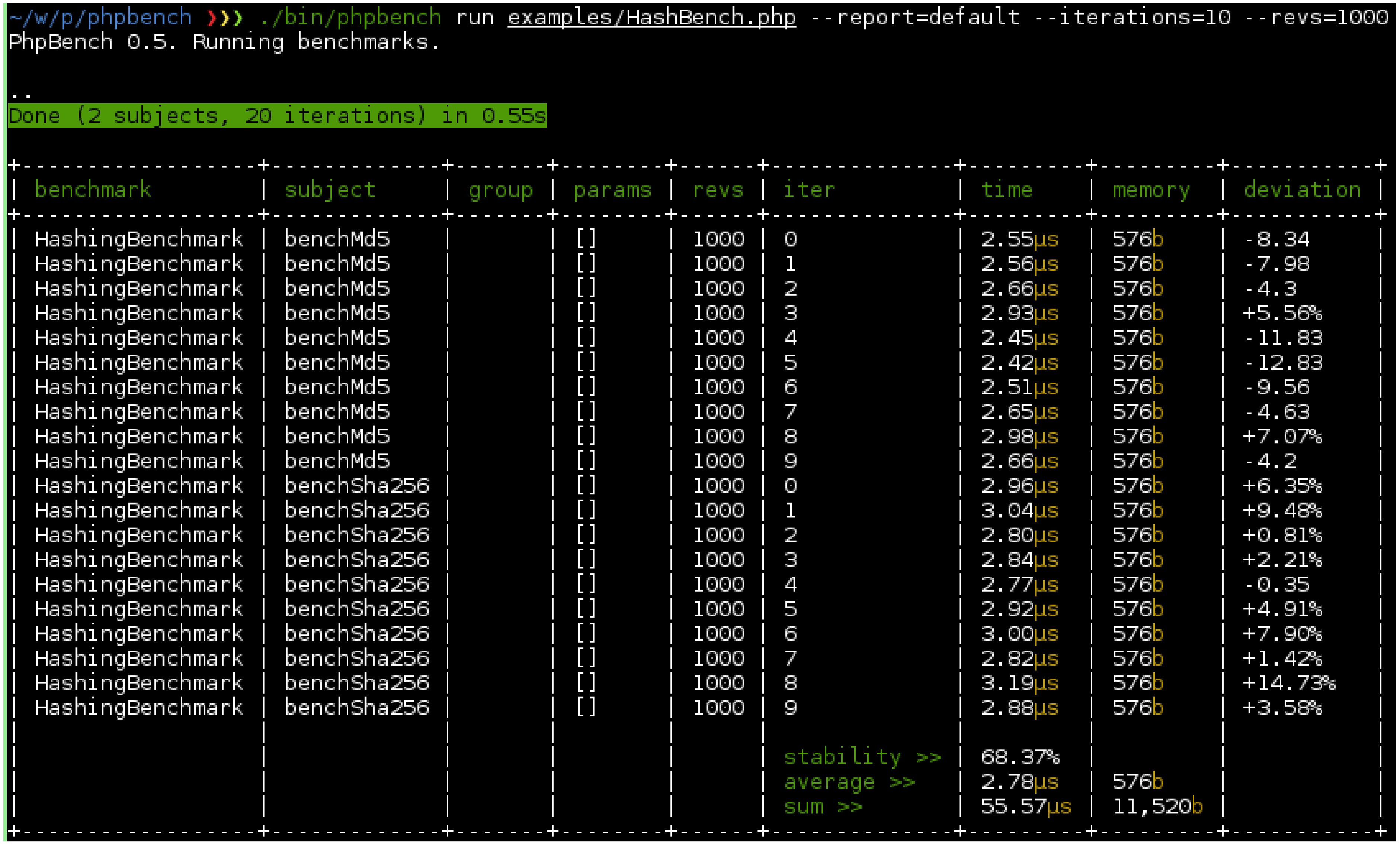
- Can be named and defined in the PHPBench configuration
- Can be specified directly on the command line:



```
~/w/p/phpbench ❯❯❯ ./bin/phpbench run examples/HashBench.php \
    --report='{"extends": "aggregate", "exclude": ["benchmark", "subject", "group", "params"]}'
PhpBench 0.5. Running benchmarks.

..
Done (2 subjects, 20 iterations) in 0.54s

+-------+-------+--------+--------+-----------+-----------+
| revs  | iters | time   | memory | deviation | stability |
+-------+-------+--------+--------+-----------+-----------+
| 10000 | 10    | 2.65μs | 616b   | 0.00%     | 86.54%    |
| 10000 | 10    | 3.10μs | 592b   | +17.16%   | 87.77%    |
+-------+-------+--------+--------+-----------+-----------+
```

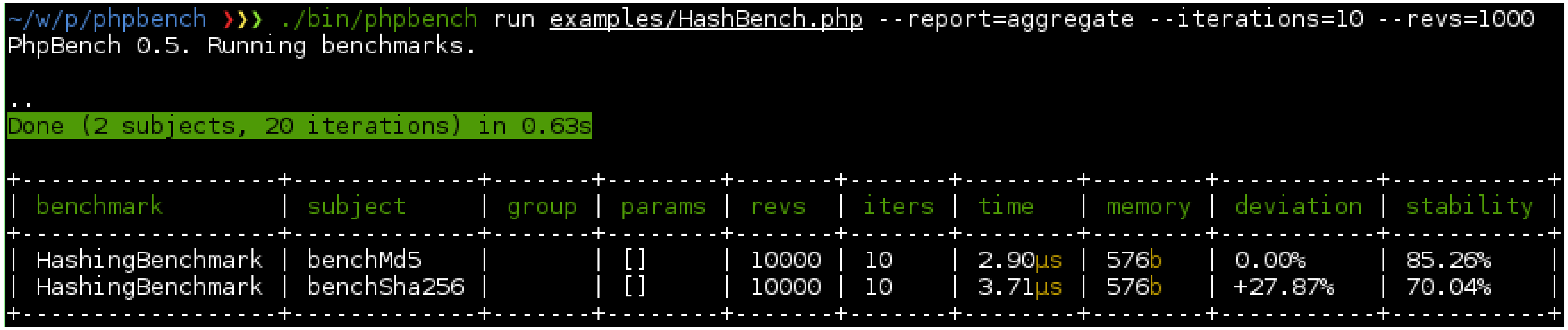
Pre-Existing Reports

Default Report



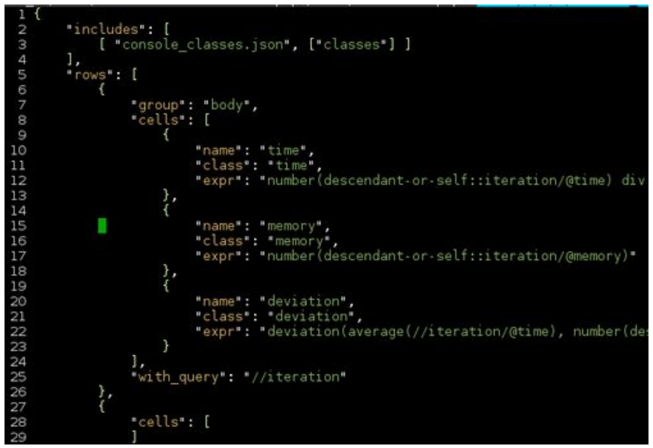
```
~/w/p/phpbench ❯❯❯ ./bin/phpbench run examples/HashBench.php --report=default --iterations=10 --revs=1000
PhpBench 0.5. Running benchmarks.

..
Done (2 subjects, 20 iterations) in 0.55s

+------------------+-------------+-------+--------+------+--------------+----------+----------+-----------+
| benchmark        | subject     | group | params | revs | iter         | time     | memory   | deviation |
+------------------+-------------+-------+--------+------+--------------+----------+----------+-----------+
| HashingBenchmark | benchMd5    |       | []     | 1000 | 0            | 2.55µs   | 576b     | -8.34     |
| HashingBenchmark | benchMd5    |       | []     | 1000 | 1            | 2.56µs   | 576b     | -7.98     |
| HashingBenchmark | benchMd5    |       | []     | 1000 | 2            | 2.66µs   | 576b     | -4.3      |
| HashingBenchmark | benchMd5    |       | []     | 1000 | 3            | 2.93µs   | 576b     | +5.56%    |
| HashingBenchmark | benchMd5    |       | []     | 1000 | 4            | 2.45µs   | 576b     | -11.83    |
| HashingBenchmark | benchMd5    |       | []     | 1000 | 5            | 2.42µs   | 576b     | -12.83    |
| HashingBenchmark | benchMd5    |       | []     | 1000 | 6            | 2.51µs   | 576b     | -9.56     |
| HashingBenchmark | benchMd5    |       | []     | 1000 | 7            | 2.65µs   | 576b     | -4.63     |
| HashingBenchmark | benchMd5    |       | []     | 1000 | 8            | 2.98µs   | 576b     | +7.07%    |
| HashingBenchmark | benchMd5    |       | []     | 1000 | 9            | 2.66µs   | 576b     | -4.2      |
| HashingBenchmark | benchSha256 |       | []     | 1000 | 0            | 2.96µs   | 576b     | +6.35%    |
| HashingBenchmark | benchSha256 |       | []     | 1000 | 1            | 3.04µs   | 576b     | +9.48%    |
| HashingBenchmark | benchSha256 |       | []     | 1000 | 2            | 2.80µs   | 576b     | +0.81%    |
| HashingBenchmark | benchSha256 |       | []     | 1000 | 3            | 2.84µs   | 576b     | +2.21%    |
| HashingBenchmark | benchSha256 |       | []     | 1000 | 4            | 2.77µs   | 576b     | -0.35     |
| HashingBenchmark | benchSha256 |       | []     | 1000 | 5            | 2.92µs   | 576b     | +4.91%    |
| HashingBenchmark | benchSha256 |       | []     | 1000 | 6            | 3.00µs   | 576b     | +7.90%    |
| HashingBenchmark | benchSha256 |       | []     | 1000 | 7            | 2.82µs   | 576b     | +1.42%    |
| HashingBenchmark | benchSha256 |       | []     | 1000 | 8            | 3.19µs   | 576b     | +14.73%   |
| HashingBenchmark | benchSha256 |       | []     | 1000 | 9            | 2.88µs   | 576b     | +3.58%    |
|                  |             |       |        |      |              |          |          |           |
|                  |             |       |        |      | stability >> | 68.37%   |          |           |
|                  |             |       |        |      | average >>   | 2.78µs   | 576b     |           |
|                  |             |       |        |      | sum >>       | 55.57µs  | 11,520b  |           |
+------------------+-------------+-------+--------+------+--------------+----------+----------+-----------+
```

Aggregate Report



```
~/w/p/phpbench ❯❯❯ ./bin/phpbench run examples/HashBench.php --report=aggregate --iterations=10 --revs=1000
PhpBench 0.5. Running benchmarks.

..
Done (2 subjects, 20 iterations) in 0.63s

+-----------------+------------+-------+--------+-------+-------+--------+--------+-----------+----------+
| benchmark       | subject    | group | params | revs  | iters | time   | memory | deviation | stability |
+-----------------+------------+-------+--------+-------+-------+--------+--------+-----------+----------+
| HashingBenchmark | benchMd5   |       | []     | 10000 | 10    | 2.90µs | 576b   | 0.00%     | 85.26%   |
| HashingBenchmark | benchSha256 |      | []     | 10000 | 10    | 3.71µs | 576b   | +27.87%   | 70.04%   |
+-----------------+------------+-------+--------+-------+-------+--------+--------+-----------+----------+
```

Custom Reports



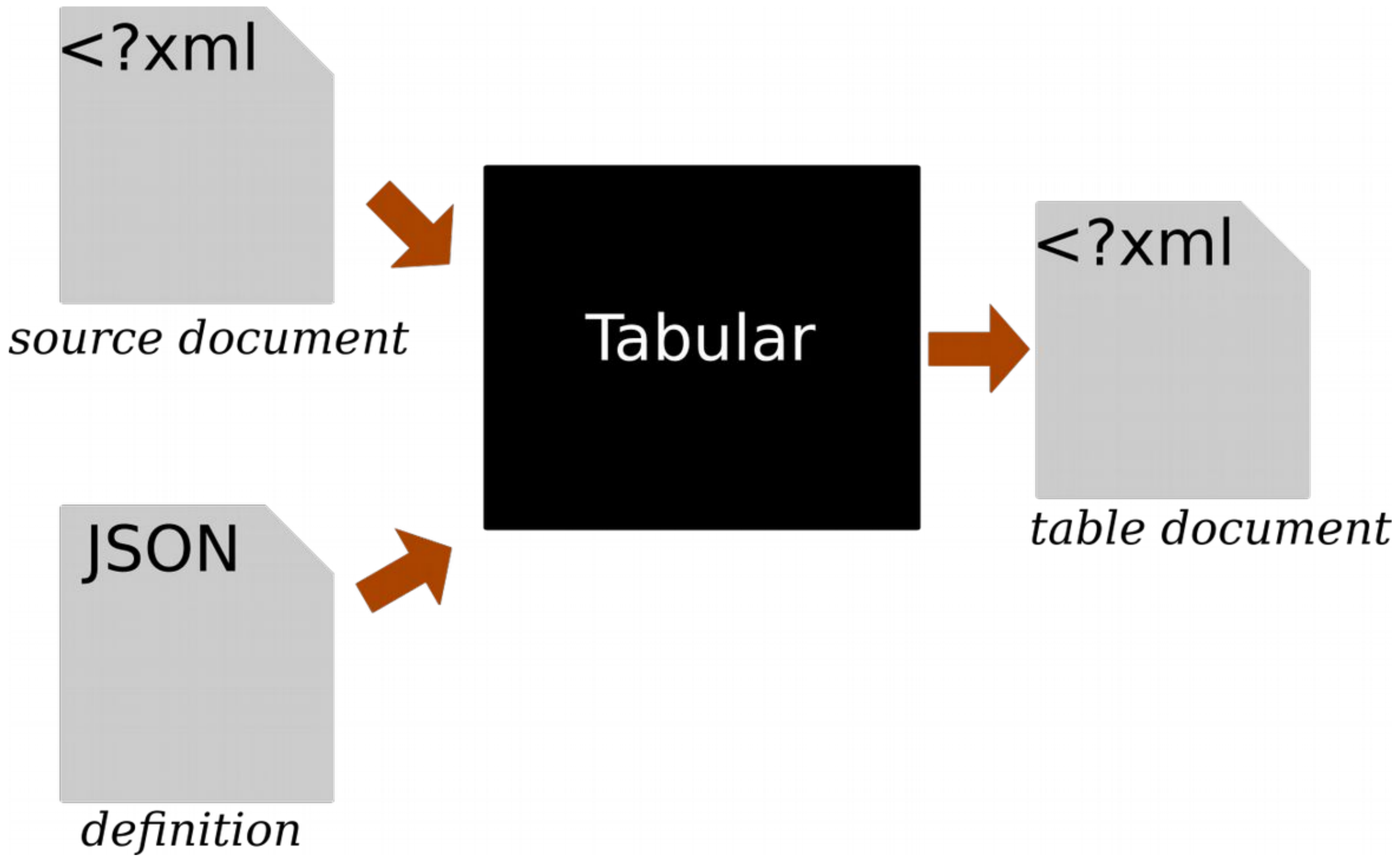
```
{
    "includes": [
        [ "console_classes.json", ["classes"] ]
    ],
    "rows": [
        {
            "group": "body",
            "cells": [
                {
                    "name": "time",
                    "class": "time",
                    "expr": "number(descendant-or-self::iteration/@time) div
                },
                {
                    "name": "memory",
                    "class": "memory",
                    "expr": "number(descendant-or-self::iteration/@memory)"
                },
                {
                    "name": "deviation",
                    "class": "deviation",
                    "expr": "deviation(average(//iteration/@time), number(de
                }
            ],
            "with_query": "//iteration"
        },
        {
            "cells": [
            ]
```

- New reports can be registered in the configuration file.
- The “console_table_custom” generator allows you to specify a custom Tabular report definition.

Tabular

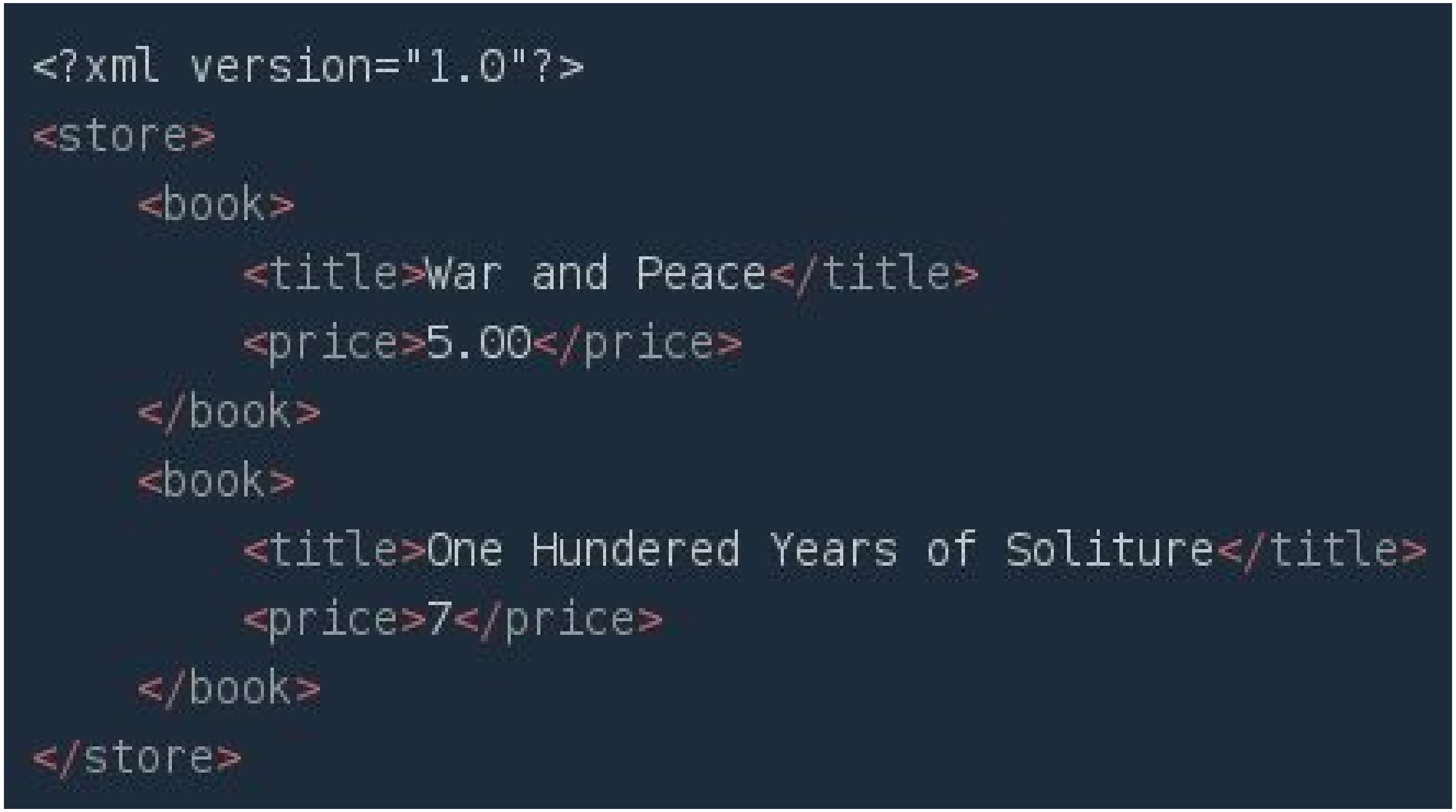
Tabular

- Developed for PHPBench but usable anywhere.
- Library for creating tabular *data*
- Accepts any XML document as its input
- Uses Xpath queries and expressions.
- Transforms it according to a Tabular JSON given definiton.



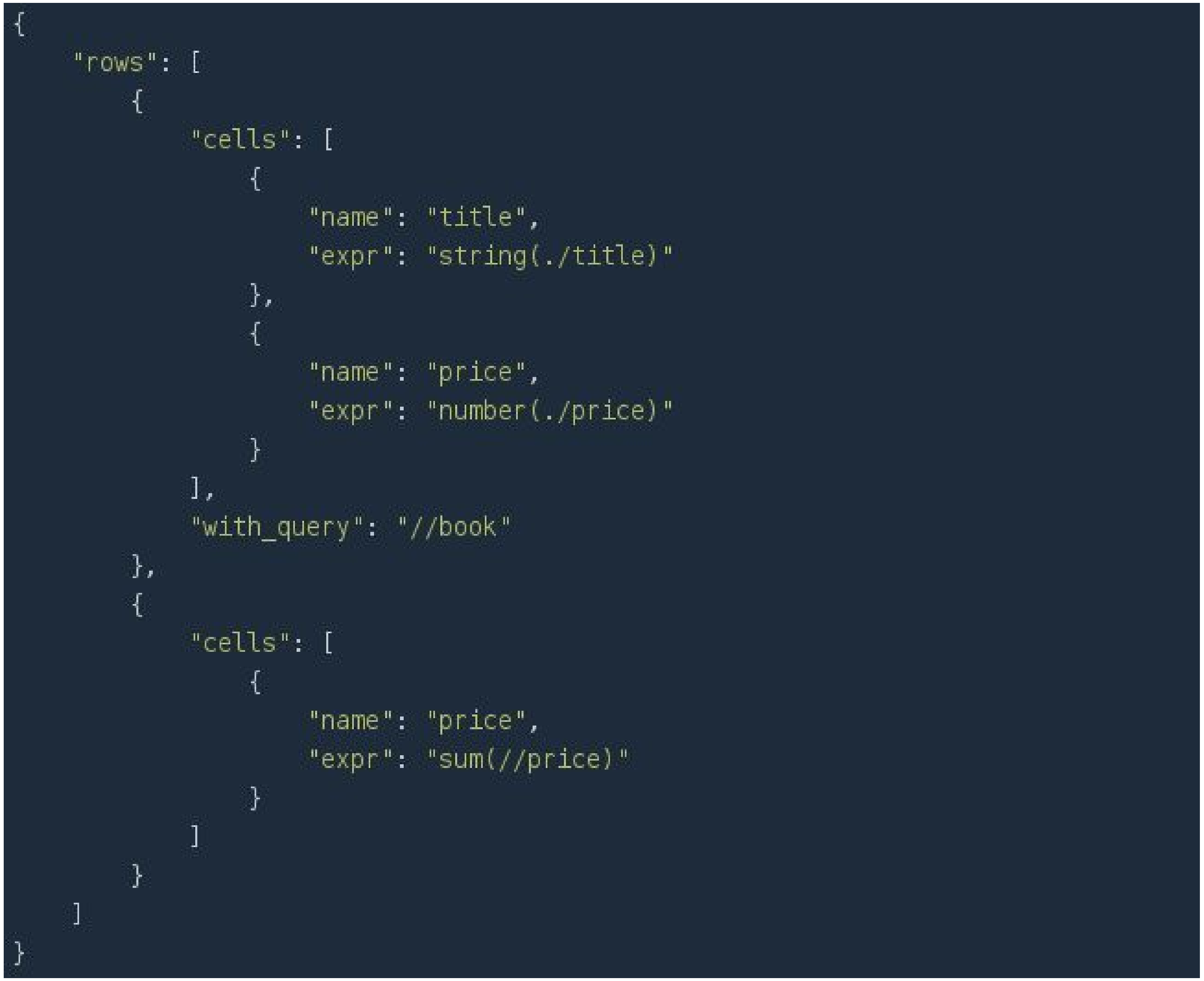
<?xml
source document
JSON
definition
Tabular
<?xml
table document

Given an XML file



```
<?xml version="1.0"?>
<store>
    <book>
        <title>War and Peace</title>
        <price>5.00</price>
    </book>
    <book>
        <title>One Hundered Years of Soliture</title>
        <price>7</price>
    </book>
</store>
```

And a Tabular definition



```
{
    "rows": [
        {
            "cells": [
                {
                    "name": "title",
                    "expr": "string(./title)"
                },
                {
                    "name": "price",
                    "expr": "number(./price)"
                }
            ],
            "with_query": "//book"
        },
        {
            "cells": [
                {
                    "name": "price",
                    "expr": "sum(//price)"
                }
            ]
        }
    ]
}
```

It generates data

Book	Price
War and Peace	5
One Hundred Years of Solitude	7
	12

- Generated data returned as an XML document
- Does not render tables

It does other stuff!

- Formats cells
- Iterate over Xpath queries
- Iterate over items
- “Compiler” passes
- Split tables into distinct groups (e.g. header, body, footer).
- Sorting
- Include other definitions

Conclusion

- Benchmarking can help you make smart decisions.
- PHPBench can be used to obtain more *reliable* benchmark results.

Questions?

Live Coding!

“This could go wrong ...”

The End.

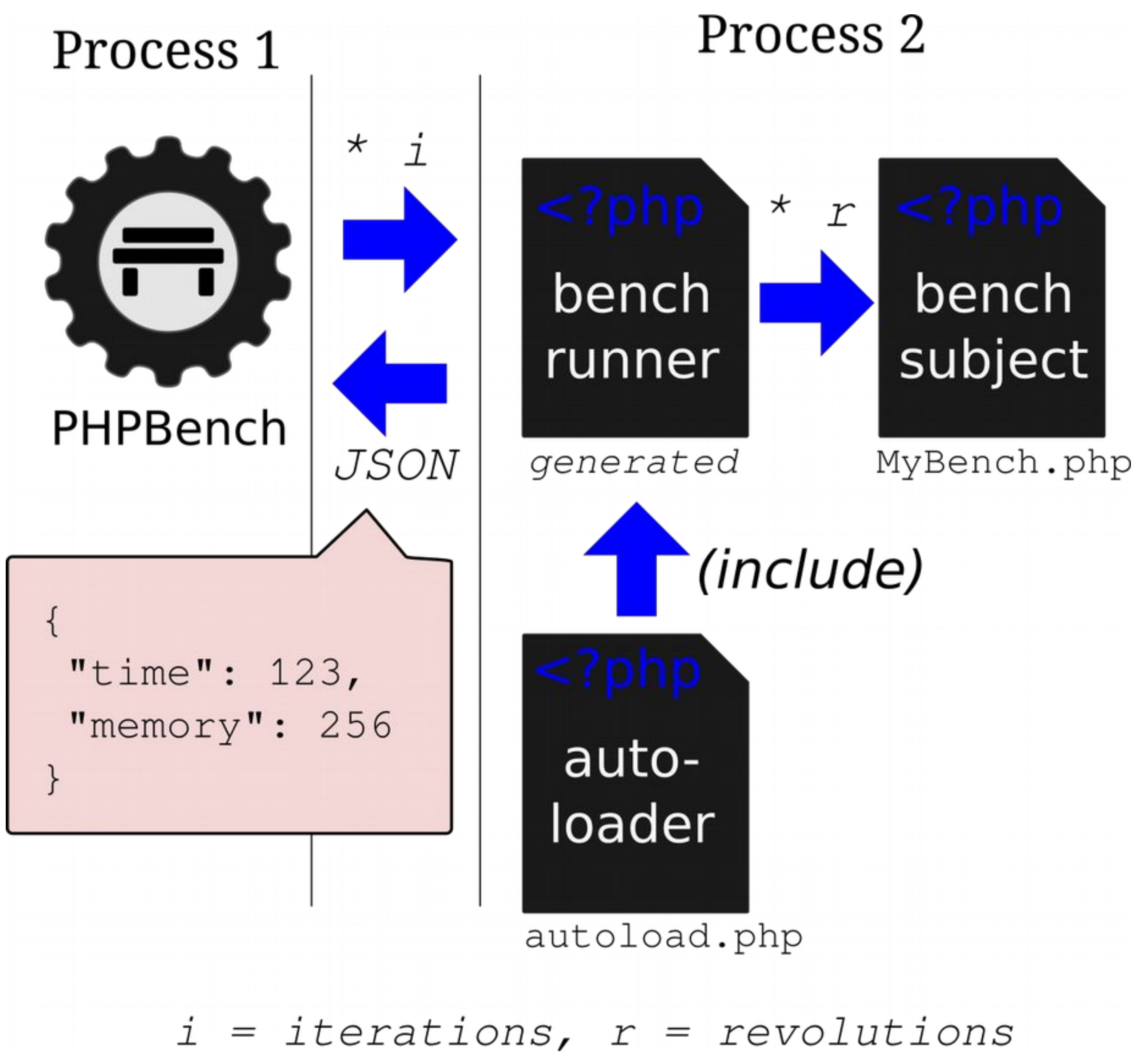
- Twitter: @dantleech
- Website: www.dantleech.com
- PHPBench: github.com/phpbench/phpbench
- Tabular: github.com/phpbench/tabular

Extra

Benchmark Process Isolation

- Benchmarks executed by a generated script in a separate process.
- PHPBench does add additional overhead to your benchmarks.

How it works

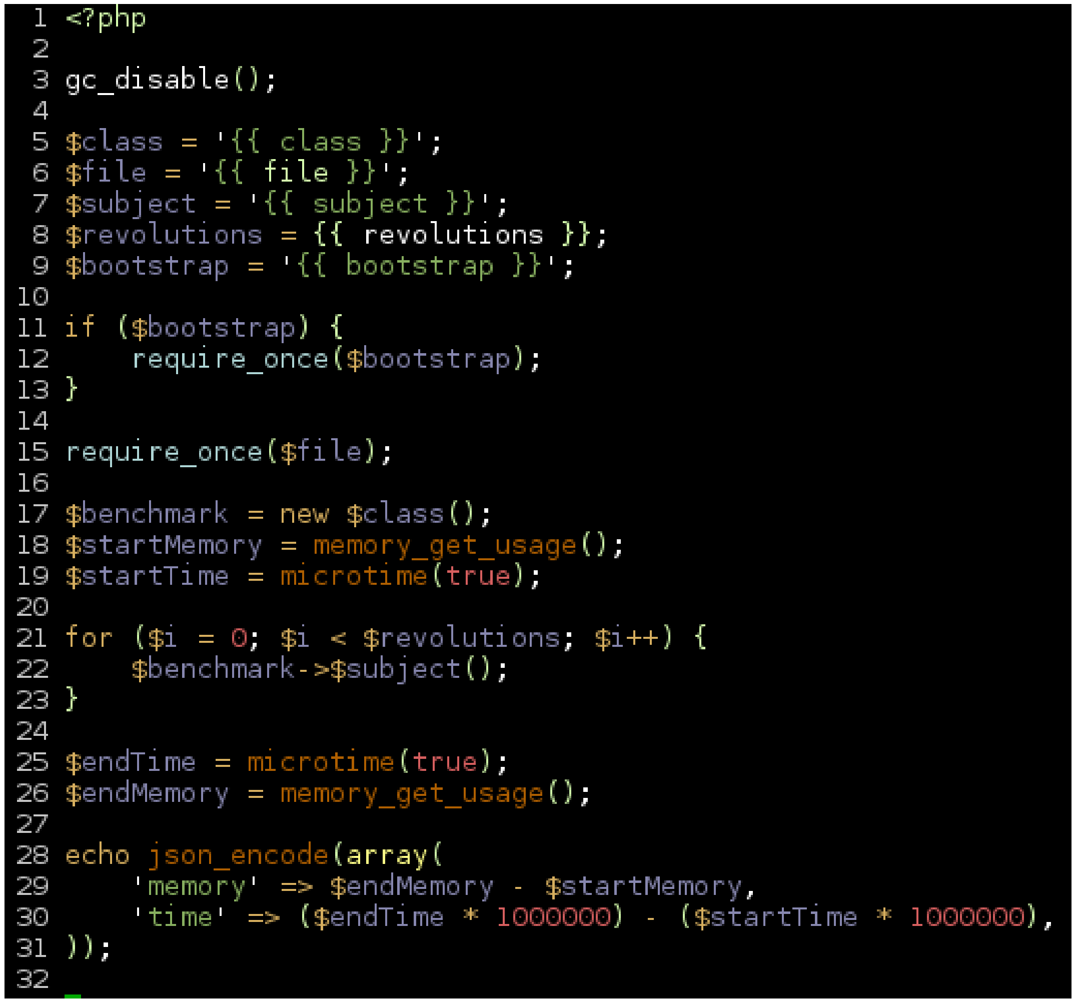


```
{
 "time": 123,
 "memory": 256
}
```

The Benchmarking Script

- Gererated in /tmp
- Executed with the Symfony Process component

PHPBench Benchmarking Script



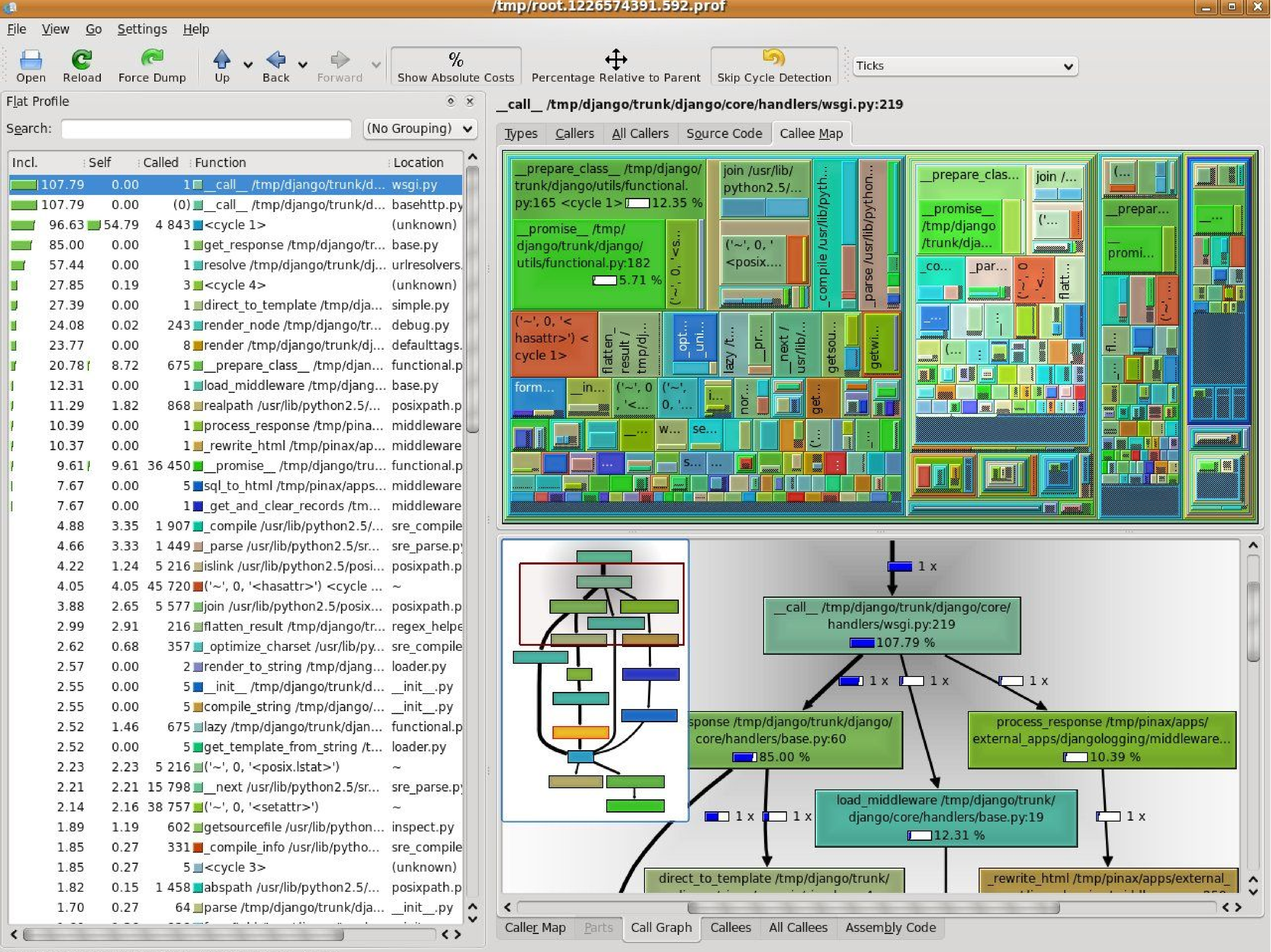
```php
<?php

gc_disable();

$class = '{{ class }}';
$file = '{{ file }}';
$subject = '{{ subject }}';
$revolutions = {{ revolutions }};
$bootstrap = '{{ bootstrap }}';

if ($bootstrap) {
    require_once($bootstrap);
}

require_once($file);

$benchmark = new $class();
$startMemory = memory_get_usage();
$startTime = microtime(true);

for ($i = 0; $i < $revolutions; $i++) {
    $benchmark->$subject();
}

$endTime = microtime(true);
$endMemory = memory_get_usage();

echo json_encode(array(
    'memory' => $endMemory - $startMemory,
    'time' => ($endTime * 1000000) - ($startTime * 1000000),
));

```

script template (simplified)

- Does not require PHPBench to be autoloaded
- Generated in tmp directory
- Disables Garbage Collection

What About Profiling?

- Provides detailed analaysis of the whole lifecycle, including timings
- Essential for a deeper understanding of code performance
- Tools
  - Xdebug with a visualiser (e.g. KCacheGrind)
  - Blackfire.io


/tmp/root.1226574391.592.prof
File View Go Settings Help
Open Reload Force Dump Up Back Forward Show Absolute Costs Percentage Relative to Parent Skip Cycle Detection
Ticks
Flat Profile
Search:
(No Grouping)
__call__ /tmp/django/trunk/django/core/handlers/wsgi.py:219
Types Callers All Callers Source Code Callee Map
__prepare_class__ /tmp/django/trunk/django/utils/functional.py:165 <cycle 1> 12.35 %
__promise__ /tmp/django/trunk/django/utils/functional.py:182 5.71 %
join /usr/lib/python2.5/...
('~', 0, '<hasattr>') < cycle 1>
1 x
__call__ /tmp/django/trunk/django/core/handlers/wsgi.py:219 107.79 %
sponse /tmp/django/trunk/django/core/handlers/base.py:60 85.00 %
process_response /tmp/pinax/apps/external_apps/djangologging/middleware... 10.39 %
load_middleware /tmp/django/trunk/django/core/handlers/base.py:19 12.31 %
direct_to_template /tmp/django/trunk/
_rewrite_html /tmp/pinax/apps/external_
Caller Map Parts Call Graph Callees All Callees Assembly Code

Incl.	Self	Called	Function	Location
107.79	0.00	1	__call__ /tmp/django/trunk/d...	wsgi.py
107.79	0.00	(0)	__call__ /tmp/django/trunk/d...	basehttp.py
96.63	54.79	4 843	<cycle 1>	(unknown)
85.00	0.00	1	get_response /tmp/django/tr...	base.py
57.44	0.00	1	resolve /tmp/django/trunk/dj...	urlresolvers.
27.85	0.19	3	<cycle 4>	(unknown)
27.39	0.00	1	direct_to_template /tmp/dja...	simple.py
24.08	0.02	243	render_node /tmp/django/tr...	debug.py
23.77	0.00	8	render /tmp/django/trunk/dj...	defaulttags.
20.78	8.72	675	__prepare_class__ /tmp/djan...	functional.p
12.31	0.00	1	load_middleware /tmp/djang...	base.py
11.29	1.82	868	realpath /usr/lib/python2.5/...	posixpath.p
10.39	0.00	1	process_response /tmp/pina...	middleware
10.37	0.00	1	_rewrite_html /tmp/pinax/ap...	middleware
9.61	9.61	36 450	__promise__ /tmp/django/tru...	functional.p
7.67	0.00	5	sql_to_html /tmp/pinax/apps...	middleware
7.67	0.00	1	_get_and_clear_records /tm...	middleware
4.88	3.35	1 907	_compile /usr/lib/python2.5/...	sre_compile
4.66	3.33	1 449	_parse /usr/lib/python2.5/sr...	sre_parse.py
4.22	1.24	5 216	islink /usr/lib/python2.5/posi...	posixpath.p
4.05	4.05	45 720	('~', 0, '<hasattr>') <cycle ...	~
3.88	2.65	5 577	join /usr/lib/python2.5/posix...	posixpath.p
2.99	2.91	216	flatten_result /tmp/django/tr...	regex_helpe
2.62	0.68	357	_optimize_charset /usr/lib/py...	sre_compile
2.57	0.00	2	render_to_string /tmp/djang...	loader.py
2.55	0.00	5	__init__ /tmp/django/trunk/d...	__init__.py
2.55	0.00	5	compile_string /tmp/django/...	__init__.py
2.52	1.46	675	lazy /tmp/django/trunk/djan...	functional.p
2.52	0.00	5	get_template_from_string /t...	loader.py
2.23	2.23	5 216	('~', 0, '<posix.lstat>')	~
2.21	2.21	15 798	__next /usr/lib/python2.5/sr...	sre_parse.py
2.14	2.16	38 757	('~', 0, '<setattr>')	~
1.89	1.19	602	getsourcefile /usr/lib/python...	inspect.py
1.85	0.27	331	_compile_info /usr/lib/pytho...	sre_compile
1.85	0.27	5	<cycle 3>	(unknown)
1.82	0.15	1 458	abspath /usr/lib/python2.5/...	posixpath.p
1.70	0.27	64	parse /tmp/django/trunk/dja...	__init__.py

Profiling vs. Benchmarking

- Profiling
  - Is necessarily slower.
  - Mesaurements are the result of a single execution
  - Feedback is not instantaneous
- Benchmarking
  - Code runs at its natural speed
  - Units and Services can be timed in isolation
  - Feedback available instantaneously

Continuous Benchmarking?

- Automated performance regression testing
- Would require a VM that runs at a constant speed.
- Travis CI does not currently meet this requirement.

Calculating Stablity

#	Time
1	1
2	1
3	1

= 100% stability

#	Time
1	1
2	0.5
3	1

= 50% stability

Improving Stability

- Multiple iterations show correlations in samples.
- We can remove the sample with a deviation > a given threshold.
- We can then use the avergage of the remaining samples.