PHP OVER ERLANG

Manuel Rubio
Tech Lead Sweden @ Erlang Solutions

@MRonErlang
manuel.rubio@erlang-solutions.com

THE JOURNEY

1. What’s PHP?
2. BEAM and, why PHP?
3. PHP over Erlang (a.k.a. **ephp**)

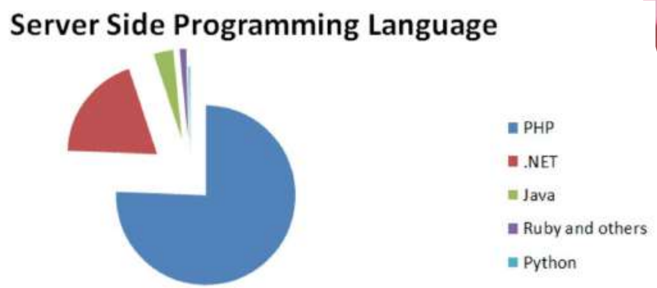
1.

WHAT'S PHP?

WHAT'S PHP?

Some features

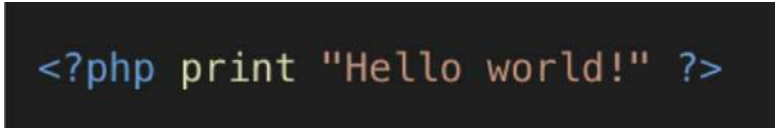
- "Accepted" syntax (yes, Java and C like)
- Easy to understand and very simple (arrays are everything)
- A lot of examples. Working examples. Some of them:
- Most of the servers in Internet run PHP:



WHAT'S PHP?

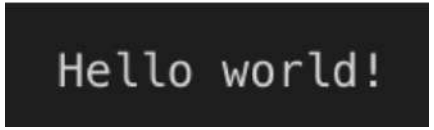
Hello world code example

You can think this is a typical "hello world!" example for PHP:



```php
<?php print "Hello world!" ?>
```

Actually, it's NOT. You can use this one instead:

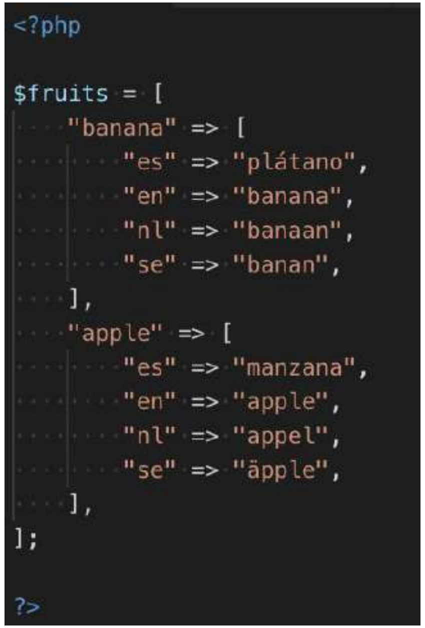


```
Hello world!
```

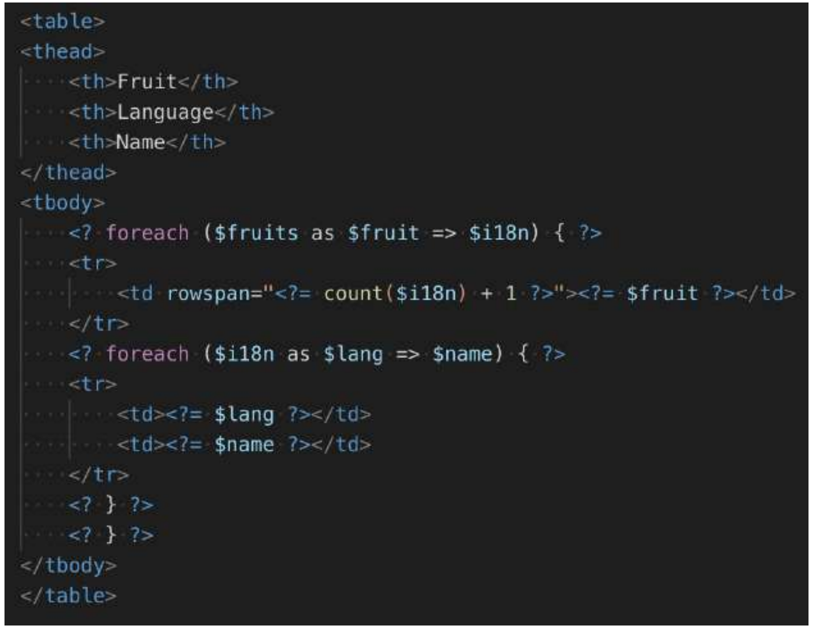
PHP is a **template language**… on steroids!

WHAT’S PHP?

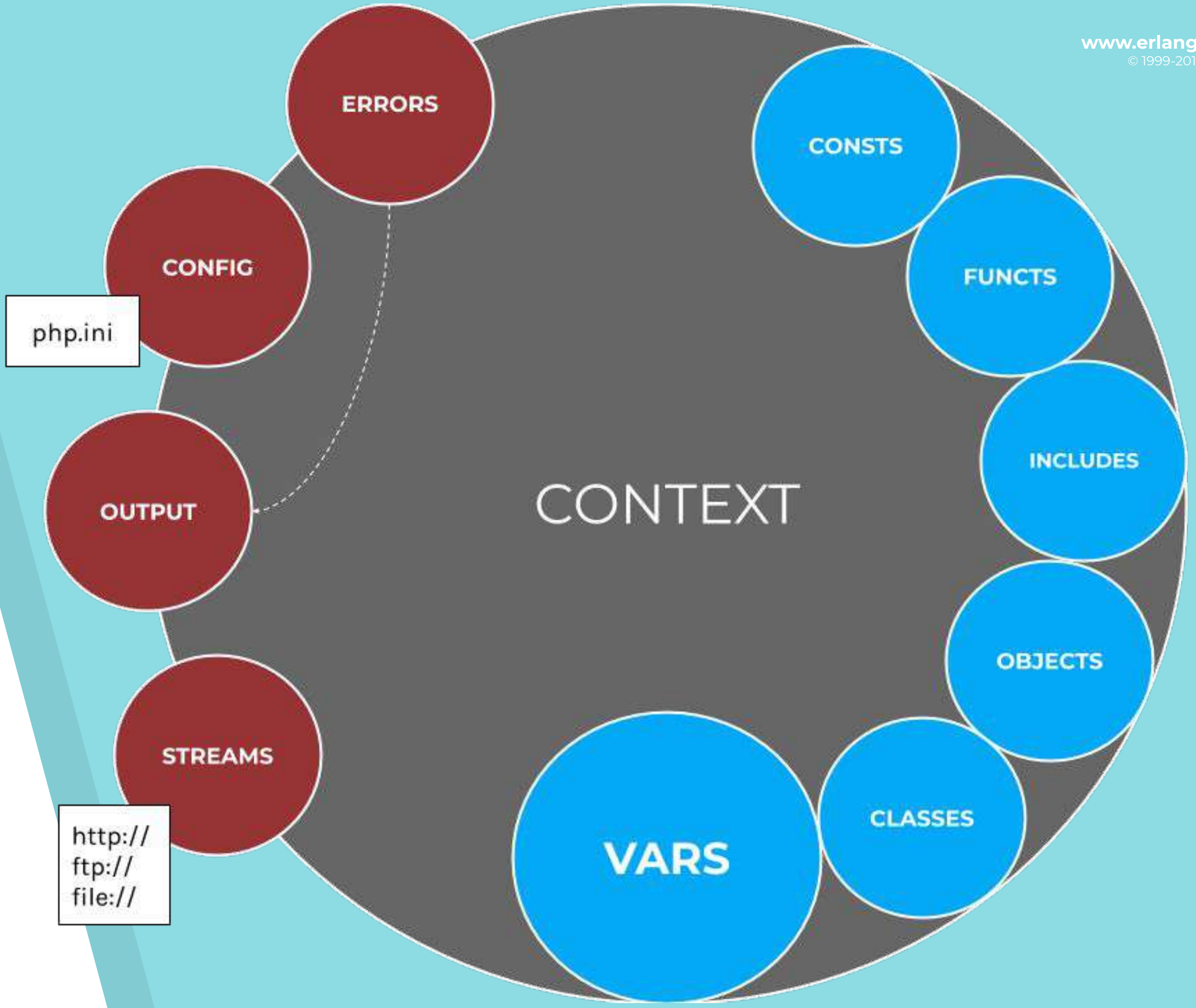
Another code example



```php
<?php

$fruits = [
    "banana" => [
        "es" => "plátano",
        "en" => "banana",
        "nl" => "banaan",
        "se" => "banan",
    ],
    "apple" => [
        "es" => "manzana",
        "en" => "apple",
        "nl" => "appel",
        "se" => "äpple",
    ],
];

?>
```



```php
<table>
<thead>
    <th>Fruit</th>
    <th>Language</th>
    <th>Name</th>
</thead>
<tbody>
    <? foreach ($fruits as $fruit => $i18n) { ?>
    <tr>
        <td rowspan="<?= count($i18n) + 1 ?>"><?= $fruit ?></td>
    </tr>
    <? foreach ($i18n as $lang => $name) { ?>
    <tr>
        <td><?= $lang ?></td>
        <td><?= $name ?></td>
    </tr>
    <? } ?>
    <? } ?>
</tbody>
</table>
```

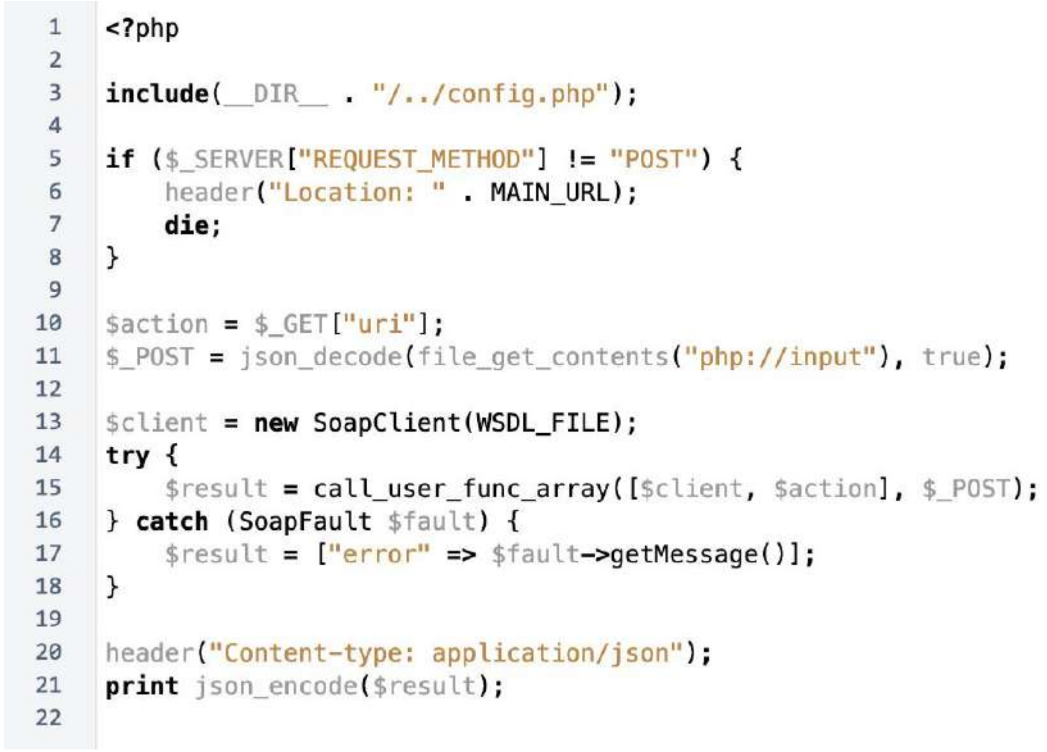


WHAT'S PHP?

Architecture

WHAT'S PHP?

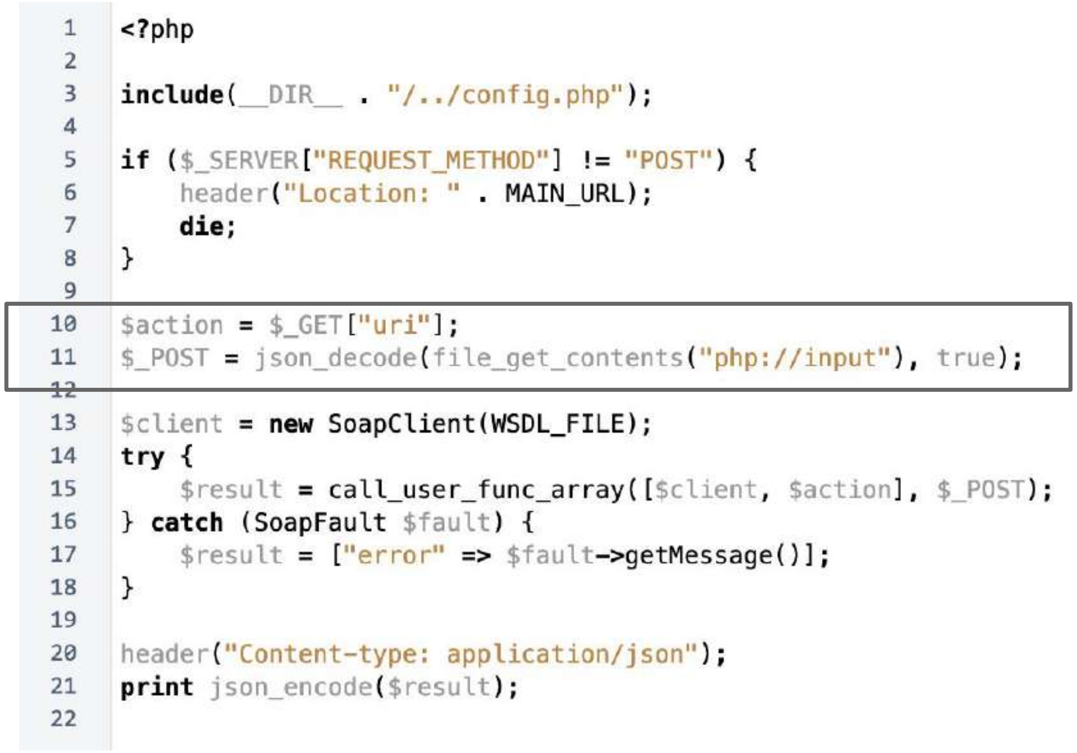
Another more code example



```php
<?php

include(__DIR__ . "/../config.php");

if ($_SERVER["REQUEST_METHOD"] != "POST") {
    header("Location: " . MAIN_URL);
    die;
}

$action = $_GET["uri"];
$_POST = json_decode(file_get_contents("php://input"), true);

$client = new SoapClient(WSDL_FILE);
try {
    $result = call_user_func_array([$client, $action], $_POST);
} catch (SoapFault $fault) {
    $result = ["error" => $fault->getMessage()];
}

header("Content-type: application/json");
print json_encode($result);

```

WHAT'S PHP?

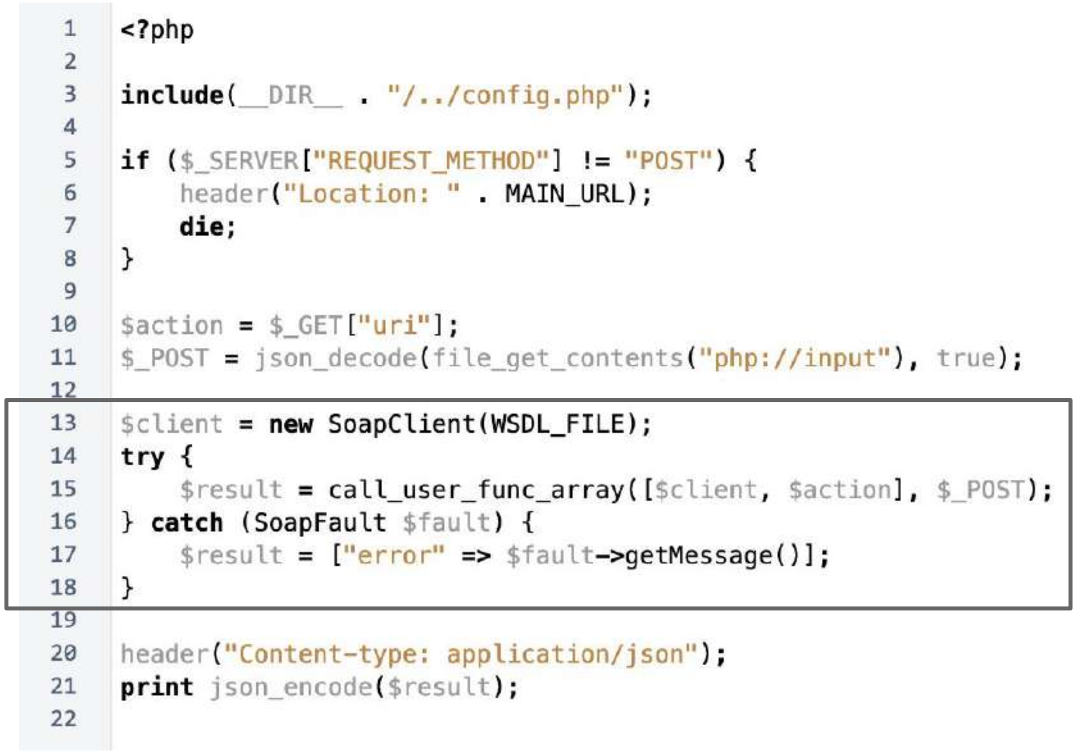
Another more code example



```php
<?php

include(__DIR__ . "/../config.php");

if ($_SERVER["REQUEST_METHOD"] != "POST") {
    header("Location: " . MAIN_URL);
    die;
}

$action = $_GET["uri"];
$_POST = json_decode(file_get_contents("php://input"), true);

$client = new SoapClient(WSDL_FILE);
try {
    $result = call_user_func_array([$client, $action], $_POST);
} catch (SoapFault $fault) {
    $result = ["error" => $fault->getMessage()];
}

header("Content-type: application/json");
print json_encode($result);

```

WHAT'S PHP?

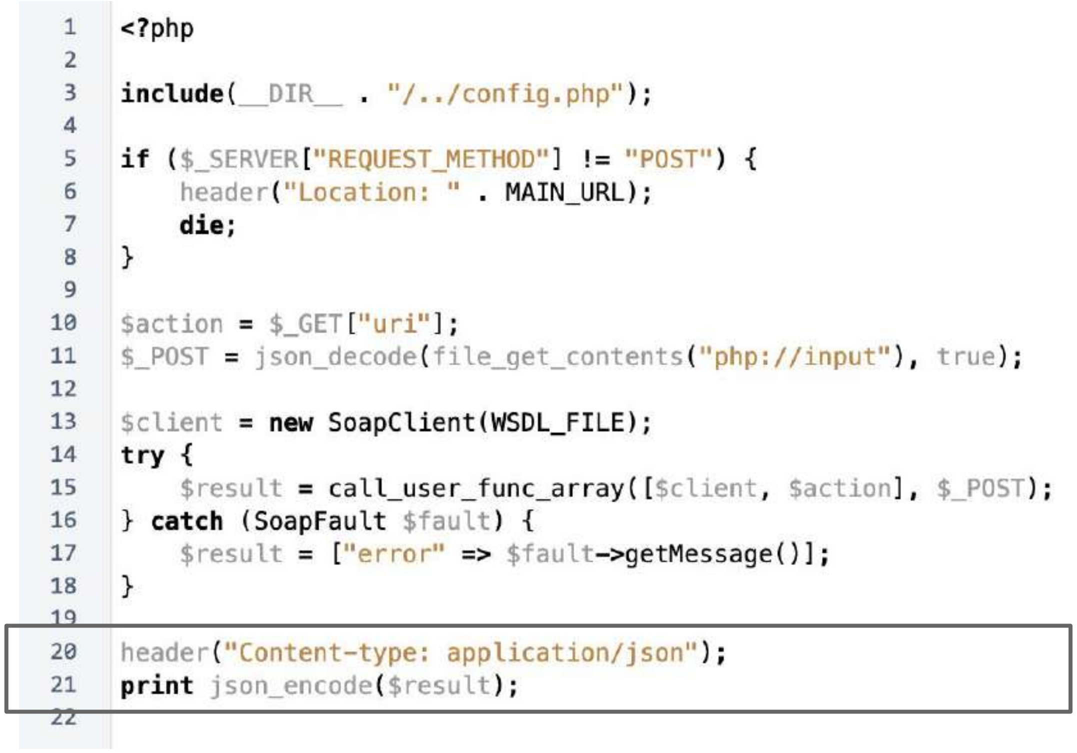
Another more code example



```php
<?php

include(__DIR__ . "/../config.php");

if ($_SERVER["REQUEST_METHOD"] != "POST") {
    header("Location: " . MAIN_URL);
    die;
}

$action = $_GET["uri"];
$_POST = json_decode(file_get_contents("php://input"), true);

$client = new SoapClient(WSDL_FILE);
try {
    $result = call_user_func_array([$client, $action], $_POST);
} catch (SoapFault $fault) {
    $result = ["error" => $fault->getMessage()];
}

header("Content-type: application/json");
print json_encode($result);
```

WHAT'S PHP?

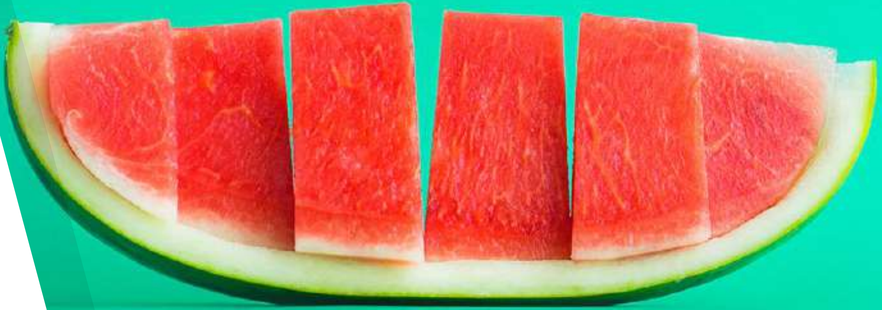
Another more code example



```php
<?php

include(__DIR__ . "/../config.php");

if ($_SERVER["REQUEST_METHOD"] != "POST") {
    header("Location: " . MAIN_URL);
    die;
}

$action = $_GET["uri"];
$_POST = json_decode(file_get_contents("php://input"), true);

$client = new SoapClient(WSDL_FILE);
try {
    $result = call_user_func_array([$client, $action], $_POST);
} catch (SoapFault $fault) {
    $result = ["error" => $fault->getMessage()];
}

header("Content-type: application/json");
print json_encode($result);

```

2.

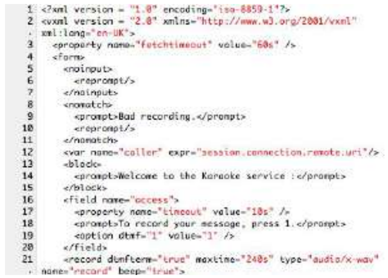
BEAM AND WHY PHP?



BEAM AND WHY PHP?

Telco companies use it!

- Because of **VoiceXML**:



```
<?xml version = "1.0" encoding="iso-8859-1"?>
<vxml version = "2.0" xmlns="http://www.w3.org/2001/vxml"
xml:lang="en-UK">
  <property name="fetchtimeout" value="60s" />
  <form>
    <noinput>
      <reprompt/>
    </noinput>
    <nomatch>
      <prompt>Bad recording.</prompt>
      <reprompt/>
    </nomatch>
    <var name="caller" expr="session.connection.remote.uri"/>
    <block>
      <prompt>Welcome to the Karaoke service :</prompt>
    </block>
    <field name="access">
      <property name="timeout" value="10s" />
      <prompt>To record your message, press 1.</prompt>
      <option dtmf="1" value="1" />
    </field>
    <record dtmfterm="true" maxtime="240s" type="audio/x-wav"
name="record" beep="true">
```

- Because of integration with **Asterisk**
- Because of integration with **FreeSwitch**
- Easy to learn → Lot of developers to hire

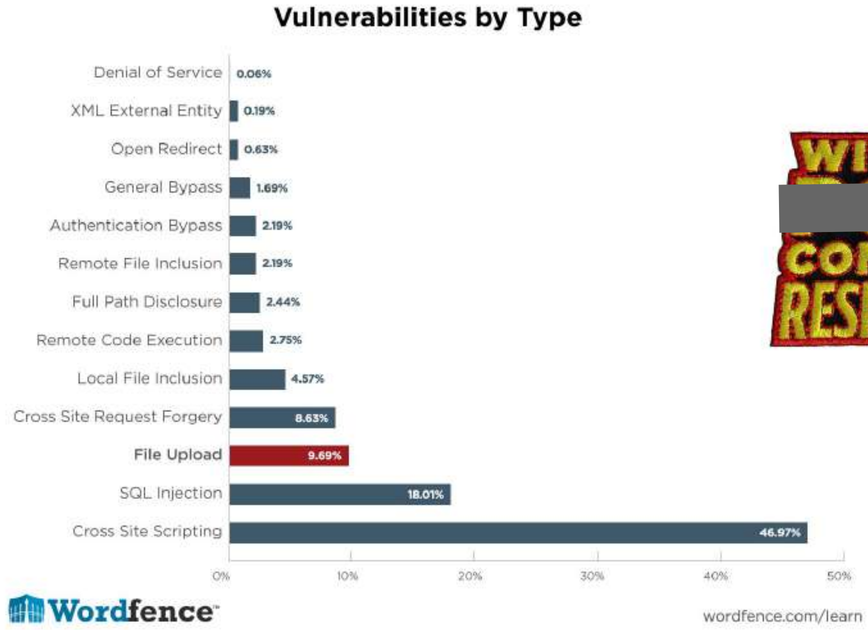
BEAM AND WHY PHP?

PHP Technology is limited…

- **No concurrency control** (**no!** Using Redis isn't concurrency development)
- **Single-thread** (no background tasks, **no!** cron isn't a good solution)
- **No real-time**, even PHP running is usually time limited
- **Limited to the Operating System** processes / threads
- **Websockets**… what's that?

BEAM AND WHY PHP?

PHP Technology is very flexible and then very insecure



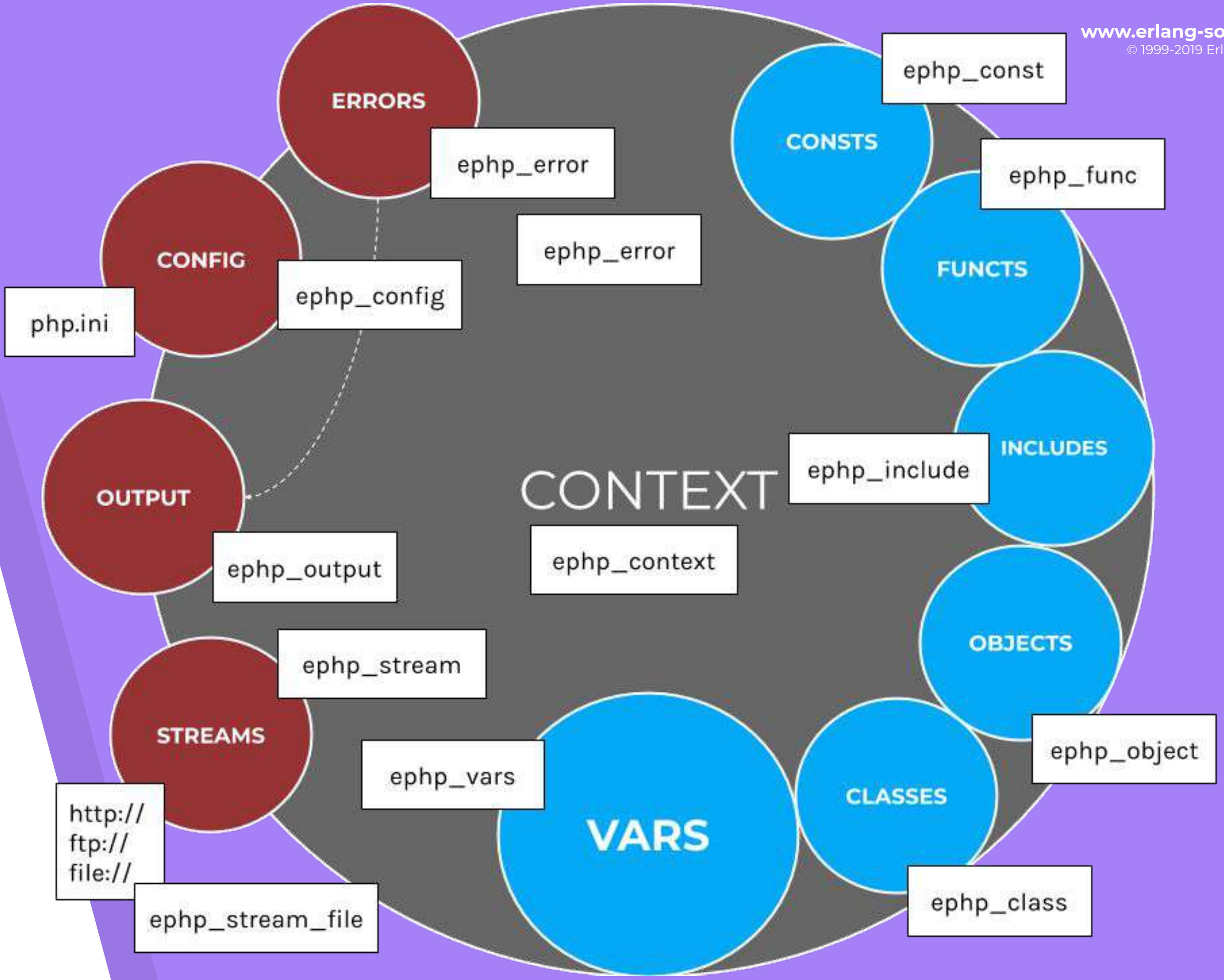
BEAM AND WHY PHP?

… then BEAM saves the day!

- **BUT** even if it's easy to learn, OTP isn't
- **BUT** it hasn't an "accepted" syntax… it's weird
- **BUT** there are no working code (maybe only Zotonic these days)
- **BUT** workarounds are better than start from scratch
- **BUT** needs to be simpler, flexible and have more examples

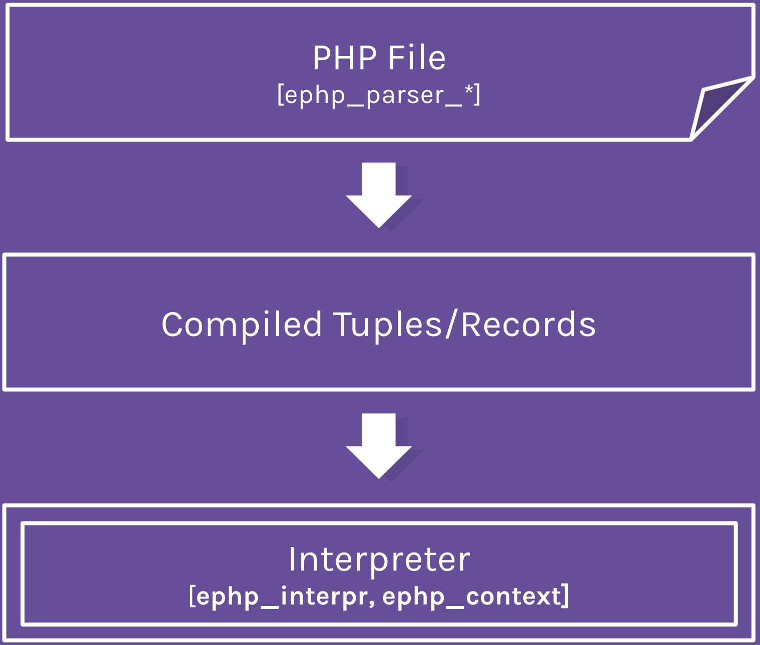
3.

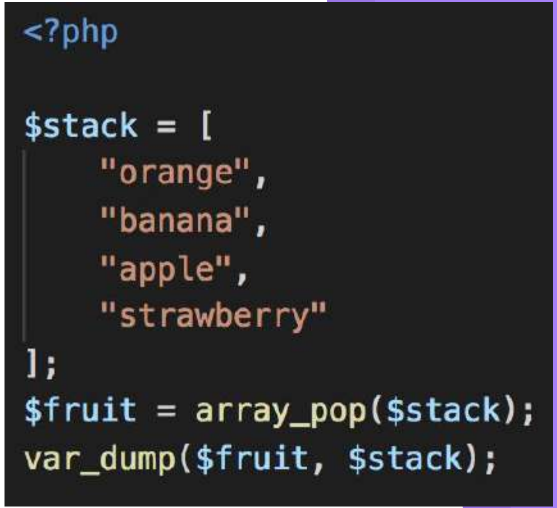
PHP over ERLANG (ephp)



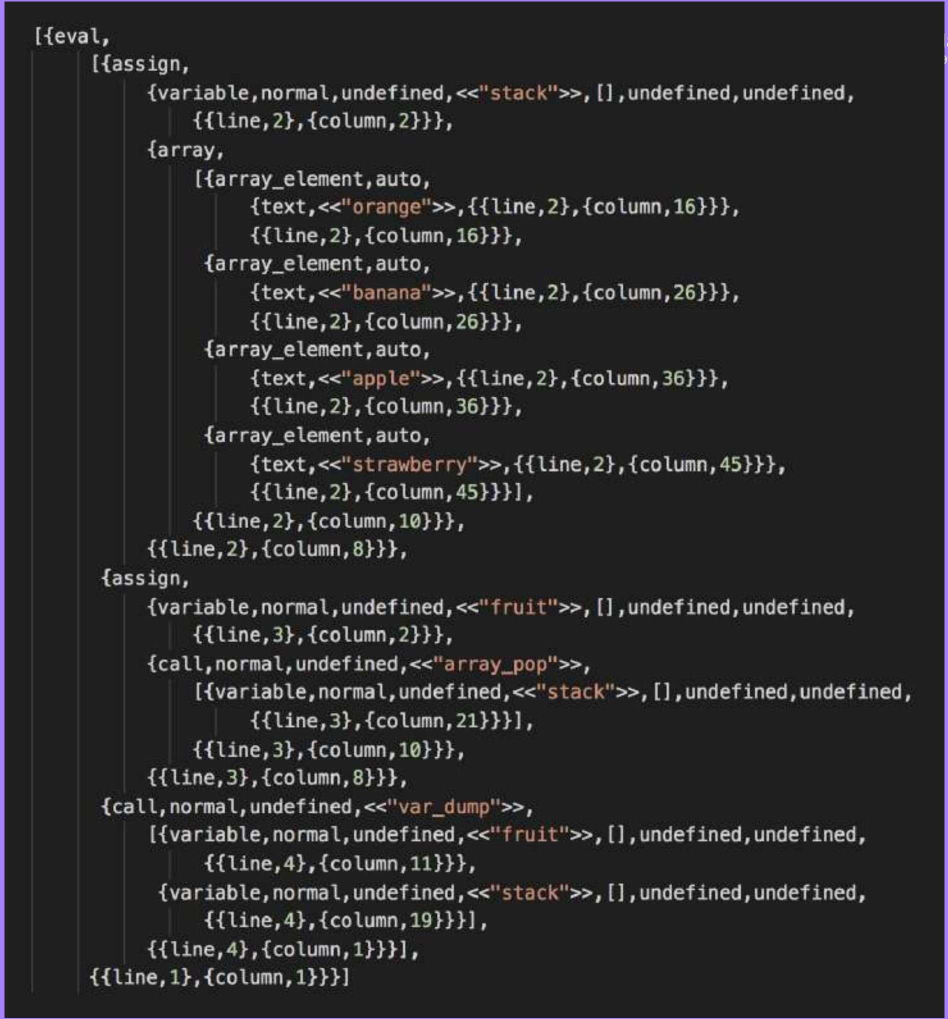
PHP over ERLANG (ephp) Architecture

PHP over ERLANG
(ephp)
Architecture





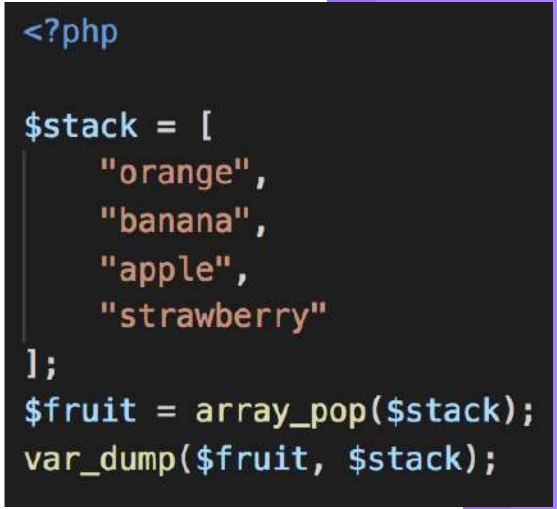
```php
<?php

$stack = [
    "orange",
    "banana",
    "apple",
    "strawberry"
];
$fruit = array_pop($stack);
var_dump($fruit, $stack);
```



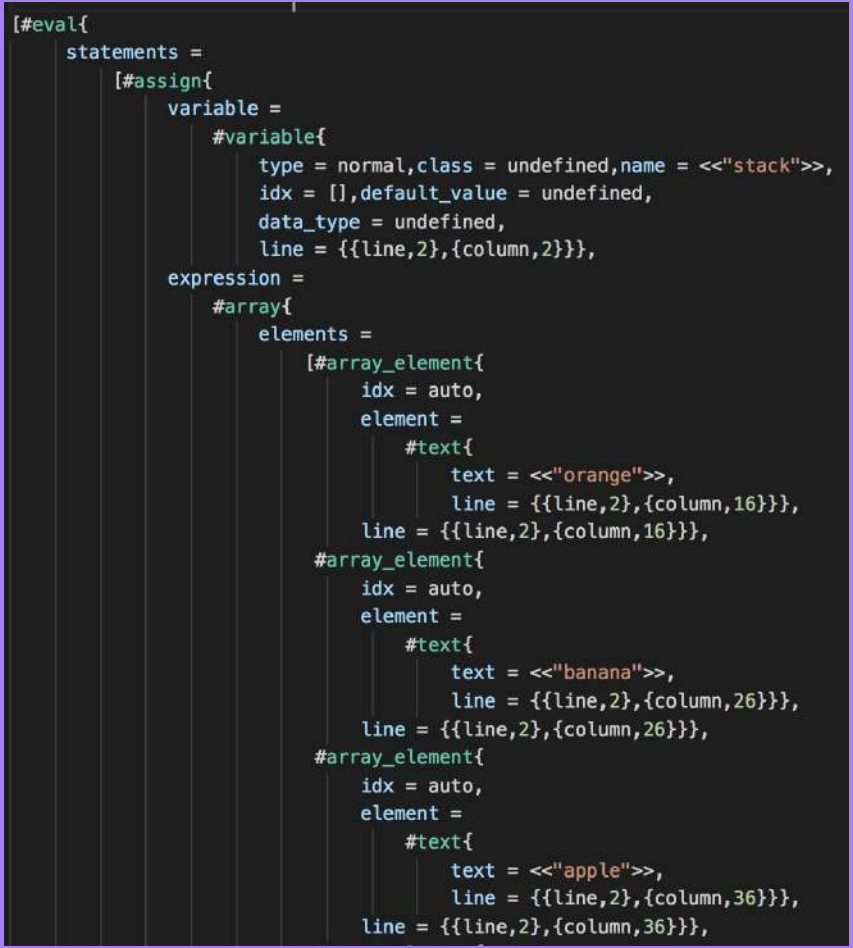
```
[{eval,
      [{assign,
            {variable,normal,undefined,<<"stack">>,[],undefined,undefined,
                {{line,2},{column,2}}},
            {array,
                [{array_element,auto,
                      {text,<<"orange">>,{{line,2},{column,16}}},
                      {{line,2},{column,16}}},
                 {array_element,auto,
                      {text,<<"banana">>,{{line,2},{column,26}}},
                      {{line,2},{column,26}}},
                 {array_element,auto,
                      {text,<<"apple">>,{{line,2},{column,36}}},
                      {{line,2},{column,36}}},
                 {array_element,auto,
                      {text,<<"strawberry">>,{{line,2},{column,45}}},
                      {{line,2},{column,45}}}],
                {{line,2},{column,10}}},
            {{line,2},{column,8}}},
       {assign,
            {variable,normal,undefined,<<"fruit">>,[],undefined,undefined,
                {{line,3},{column,2}}},
            {call,normal,undefined,<<"array_pop">>,
                [{variable,normal,undefined,<<"stack">>,[],undefined,undefined,
                      {{line,3},{column,21}}}],
                {{line,3},{column,10}}},
            {{line,3},{column,8}}},
       {call,normal,undefined,<<"var_dump">>,
             [{variable,normal,undefined,<<"fruit">>,[],undefined,undefined,
                   {{line,4},{column,11}}},
              {variable,normal,undefined,<<"stack">>,[],undefined,undefined,
                   {{line,4},{column,19}}}],
             {{line,4},{column,1}}}],
      {{line,1},{column,1}}}]
```

PHP over ERLANG (ephp)

Architecture



```php
<?php

$stack = [
    "orange",
    "banana",
    "apple",
    "strawberry"
];
$fruit = array_pop($stack);
var_dump($fruit, $stack);
```



```erlang
[#eval{
    statements =
        [#assign{
            variable =
                #variable{
                    type = normal,class = undefined,name = <<"stack">>,
                    idx = [],default_value = undefined,
                    data_type = undefined,
                    line = {{line,2},{column,2}}},
            expression =
                #array{
                    elements =
                        [#array_element{
                            idx = auto,
                            element =
                                #text{
                                    text = <<"orange">>,
                                    line = {{line,2},{column,16}}},
                            line = {{line,2},{column,16}}},
                         #array_element{
                            idx = auto,
                            element =
                                #text{
                                    text = <<"banana">>,
                                    line = {{line,2},{column,26}}},
                            line = {{line,2},{column,26}}},
                         #array_element{
                            idx = auto,
                            element =
                                #text{
                                    text = <<"apple">>,
                                    line = {{line,2},{column,36}}},
                            line = {{line,2},{column,36}}},
```

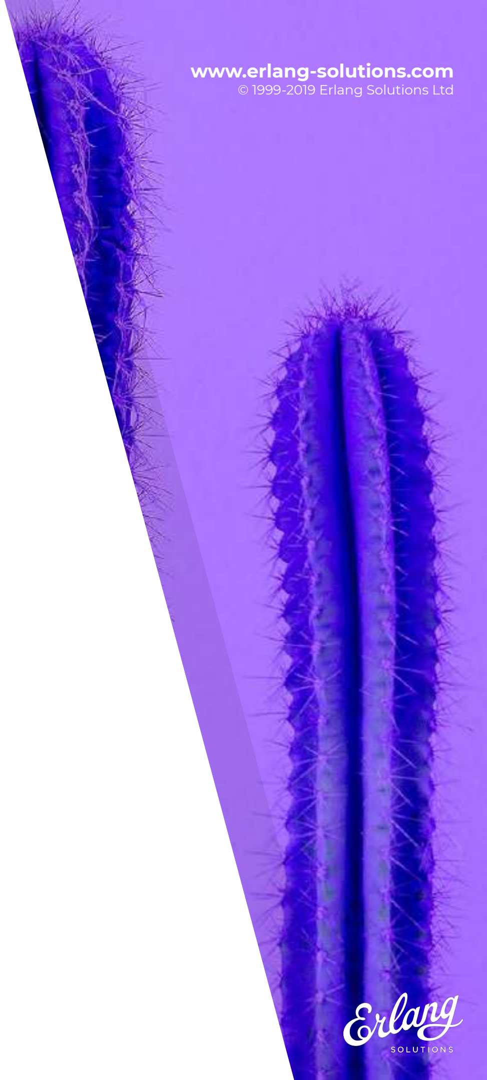
PHP over ERLANG (ephp)

Architecture

PHP over ERLANG (ephp)

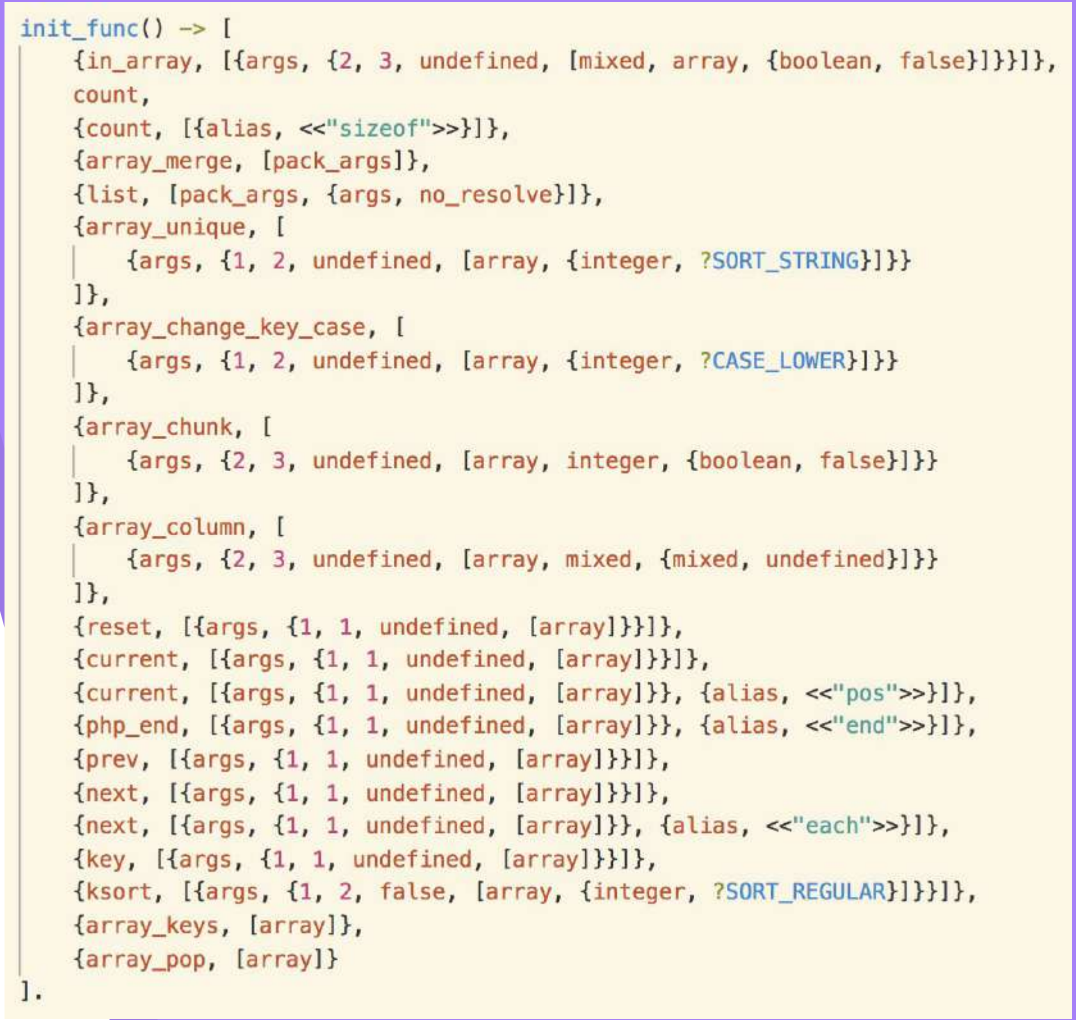
Extensibility… **ephp_func** behaviour

- **init_func** (required) list of functions to use
- **init_config** (required) configuration parameters (php.ini)
- **init_const** (required) list of constants with their values
- **handle_error** (optional) let us define new error messages
- **get_classes** (optional) list of classes added by the module

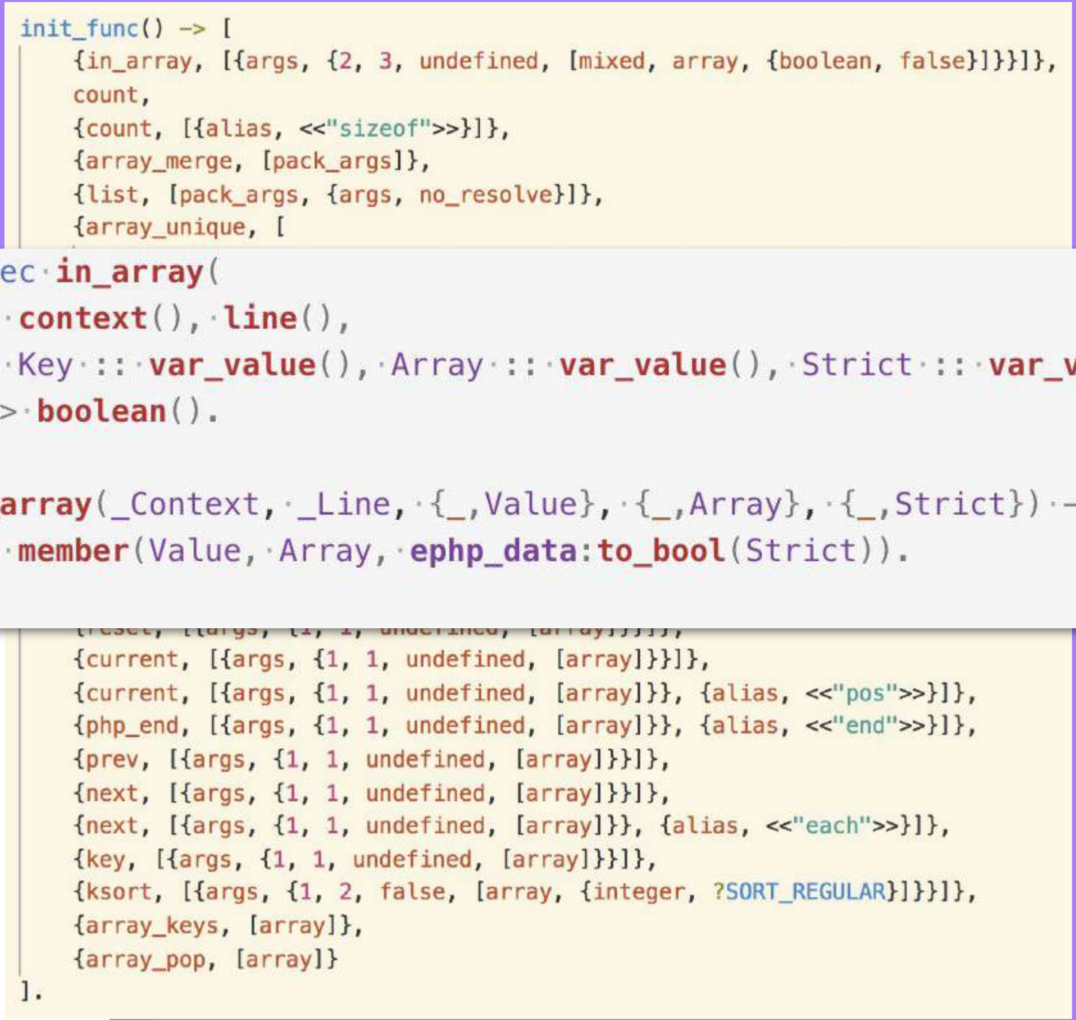


PHP over ERLANG (ephp)

Extensibility

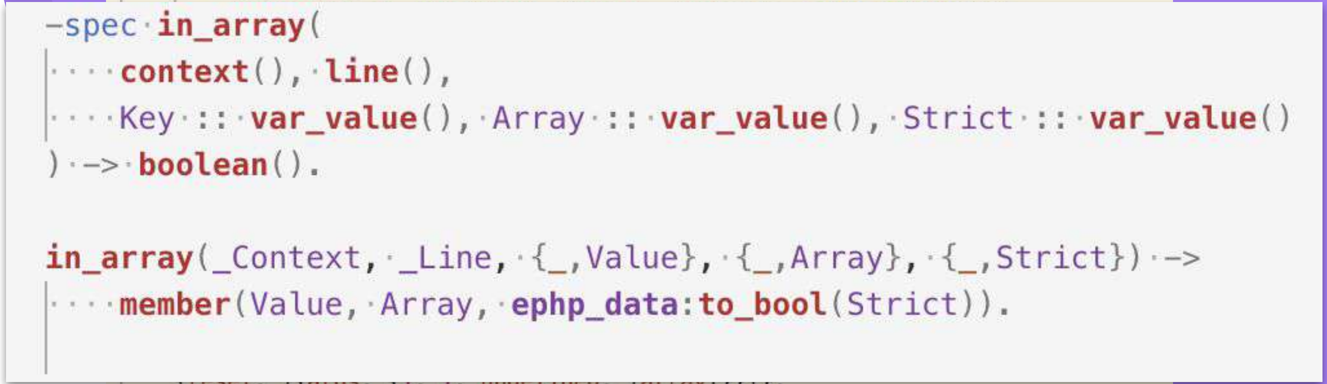


```
init_func() -> [
    {in_array, [{args, {2, 3, undefined, [mixed, array, {boolean, false}]}}]},
    count,
    {count, [{alias, <<"sizeof">>}]},
    {array_merge, [pack_args]},
    {list, [pack_args, {args, no_resolve}]},
    {array_unique, [
        {args, {1, 2, undefined, [array, {integer, ?SORT_STRING}]}}
    ]},
    {array_change_key_case, [
        {args, {1, 2, undefined, [array, {integer, ?CASE_LOWER}]}}
    ]},
    {array_chunk, [
        {args, {2, 3, undefined, [array, integer, {boolean, false}]}}
    ]},
    {array_column, [
        {args, {2, 3, undefined, [array, mixed, {mixed, undefined}]}}
    ]},
    {reset, [{args, {1, 1, undefined, [array]}}]},
    {current, [{args, {1, 1, undefined, [array]}}]},
    {current, [{args, {1, 1, undefined, [array]}}, {alias, <<"pos">>}]},
    {php_end, [{args, {1, 1, undefined, [array]}}, {alias, <<"end">>}]},
    {prev, [{args, {1, 1, undefined, [array]}}]},
    {next, [{args, {1, 1, undefined, [array]}}]},
    {next, [{args, {1, 1, undefined, [array]}}, {alias, <<"each">>}]},
    {key, [{args, {1, 1, undefined, [array]}}]},
    {ksort, [{args, {1, 2, false, [array, {integer, ?SORT_REGULAR}]}}]},
    {array_keys, [array]},
    {array_pop, [array]}
].
```



```erlang
init_func() -> [
    {in_array, [{args, {2, 3, undefined, [mixed, array, {boolean, false}]}}]},
    count,
    {count, [{alias, <<"sizeof">>}]},
    {array_merge, [pack_args]},
    {list, [pack_args, {args, no_resolve}]},
    {array_unique, [
```

```erlang
    {current, [{args, {1, 1, undefined, [array]}}]},
    {current, [{args, {1, 1, undefined, [array]}}, {alias, <<"pos">>}]},
    {php_end, [{args, {1, 1, undefined, [array]}}, {alias, <<"end">>}]},
    {prev, [{args, {1, 1, undefined, [array]}}]},
    {next, [{args, {1, 1, undefined, [array]}}]},
    {next, [{args, {1, 1, undefined, [array]}}, {alias, <<"each">>}]},
    {key, [{args, {1, 1, undefined, [array]}}]},
    {ksort, [{args, {1, 2, false, [array, {integer, ?SORT_REGULAR}]}}]},
    {array_keys, [array]},
    {array_pop, [array]}
].
```



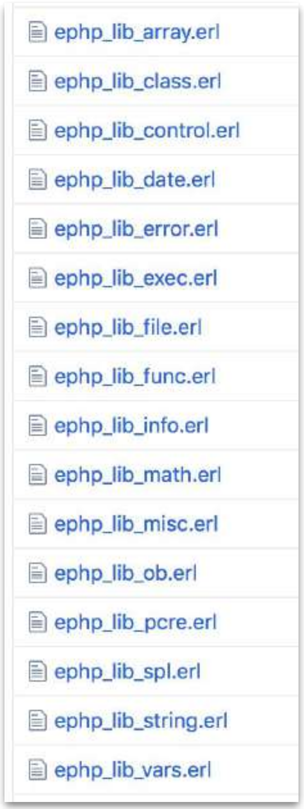
```erlang
-spec in_array(
    context(), line(),
    Key :: var_value(), Array :: var_value(), Strict :: var_value()
) -> boolean().

in_array(_Context, _Line, {_,Value}, {_,Array}, {_,Strict}) ->
    member(Value, Array, ephp_data:to_bool(Strict)).
```

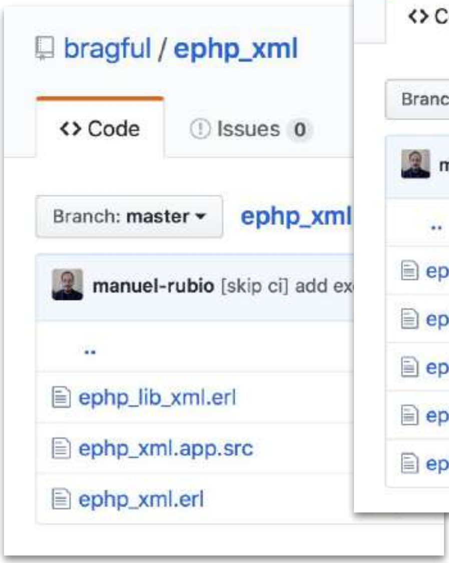
PHP over ERLANG

(ephp)

Extensibility

PHP over ERLANG (ephp)


ephp_lib_array.erl
ephp_lib_class.erl
ephp_lib_control.erl
ephp_lib_date.erl
ephp_lib_error.erl
ephp_lib_exec.erl
ephp_lib_file.erl
ephp_lib_func.erl
ephp_lib_info.erl
ephp_lib_math.erl
ephp_lib_misc.erl
ephp_lib_ob.erl
ephp_lib_pcre.erl
ephp_lib_spl.erl
ephp_lib_string.erl
ephp_lib_vars.erl

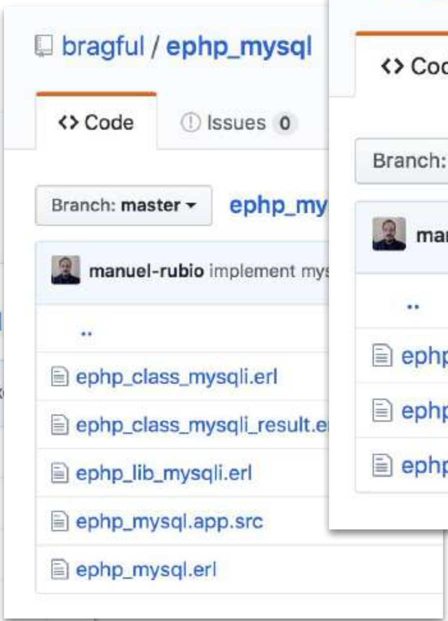

bragful / ephp_xml

<> Code | Issues 0

Branch: master ▾ ephp_xml

manuel-rubio [skip ci] add ex

..
ephp_lib_xml.erl
ephp_xml.app.src
ephp_xml.erl

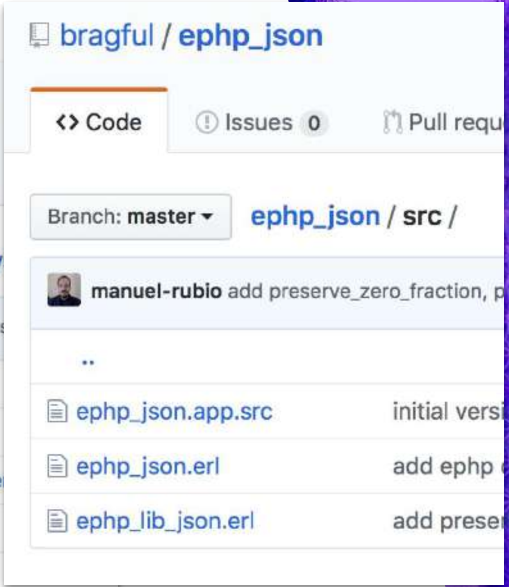

bragful / ephp_mysql

<> Code | Issues 0

Branch: master ▾ ephp_my

manuel-rubio implement mys

..
ephp_class_mysqli.erl
ephp_class_mysqli_result.e
ephp_lib_mysqli.erl
ephp_mysql.app.src
ephp_mysql.erl


bragful / ephp_json

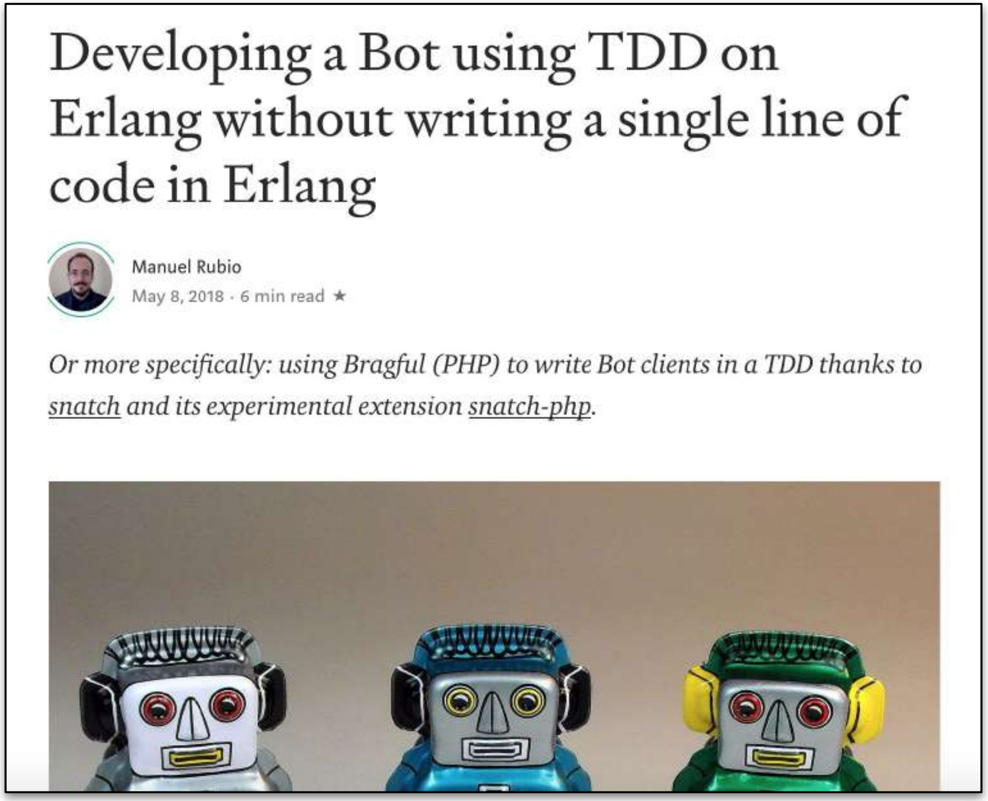
<> Code | Issues 0 | Pull requ

Branch: master ▾ ephp_json / src /

manuel-rubio add preserve_zero_fraction, p

..
ephp_json.app.src initial versi
ephp_json.erl add ephp
ephp_lib_json.erl add prese

PHP over ERLANG (ephp)

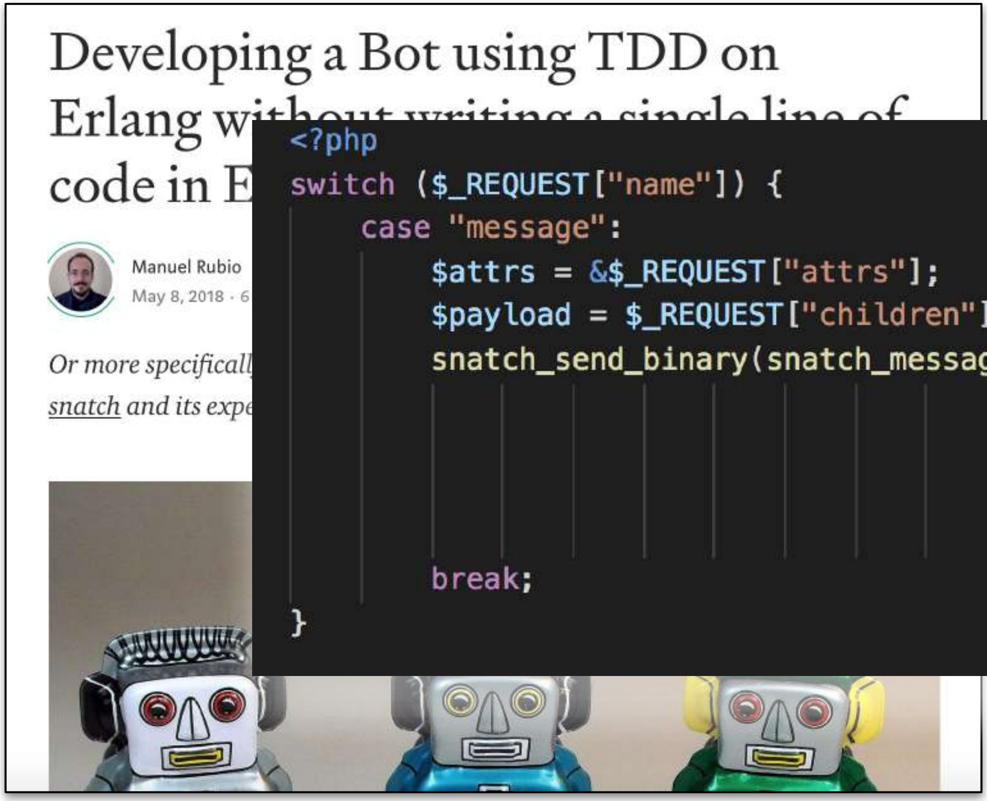


Developing a Bot using TDD on Erlang without writing a single line of code in Erlang

Manuel Rubio
May 8, 2018 · 6 min read ★

Or more specifically: using Bragful (PHP) to write Bot clients in a TDD thanks to snatch and its experimental extension snatch-php.

PHP over ERLANG (ephp)



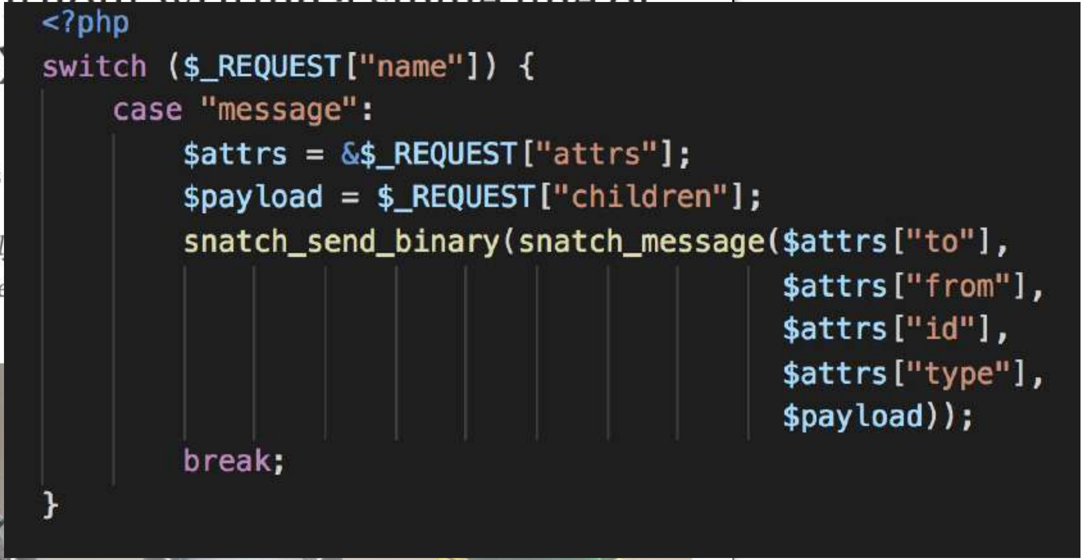
Developing a Bot using TDD on Erlang w

code in E

Manuel Rubio
May 8, 2018 · 6

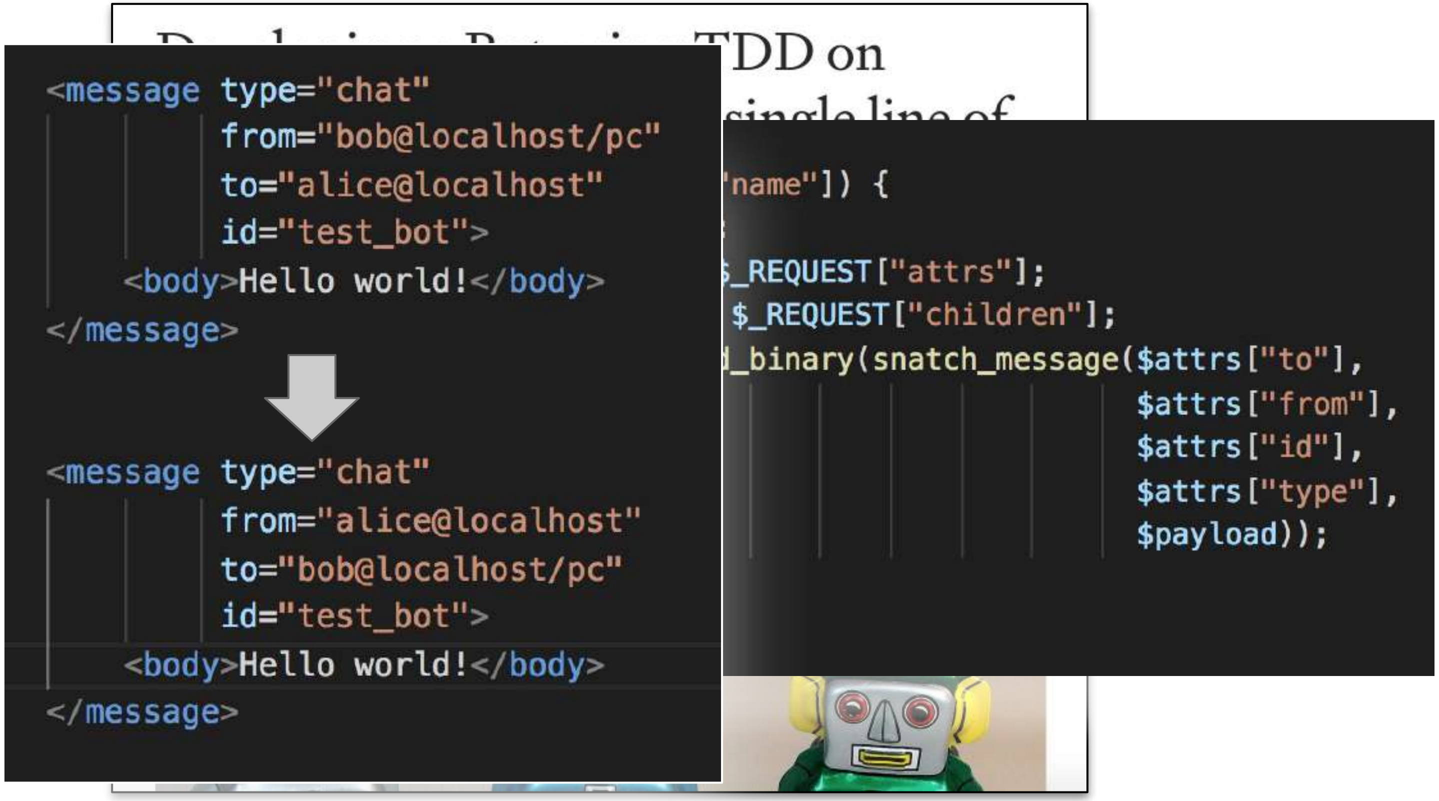
Or more specificall

snatch and its expe



```
<?php
switch ($_REQUEST["name"]) {
    case "message":
        $attrs = &$_REQUEST["attrs"];
        $payload = $_REQUEST["children"];
        snatch_send_binary(snatch_message($attrs["to"],
                                          $attrs["from"],
                                          $attrs["id"],
                                          $attrs["type"],
                                          $payload));
        break;
}
```

PHP over ERLANG (ephp)



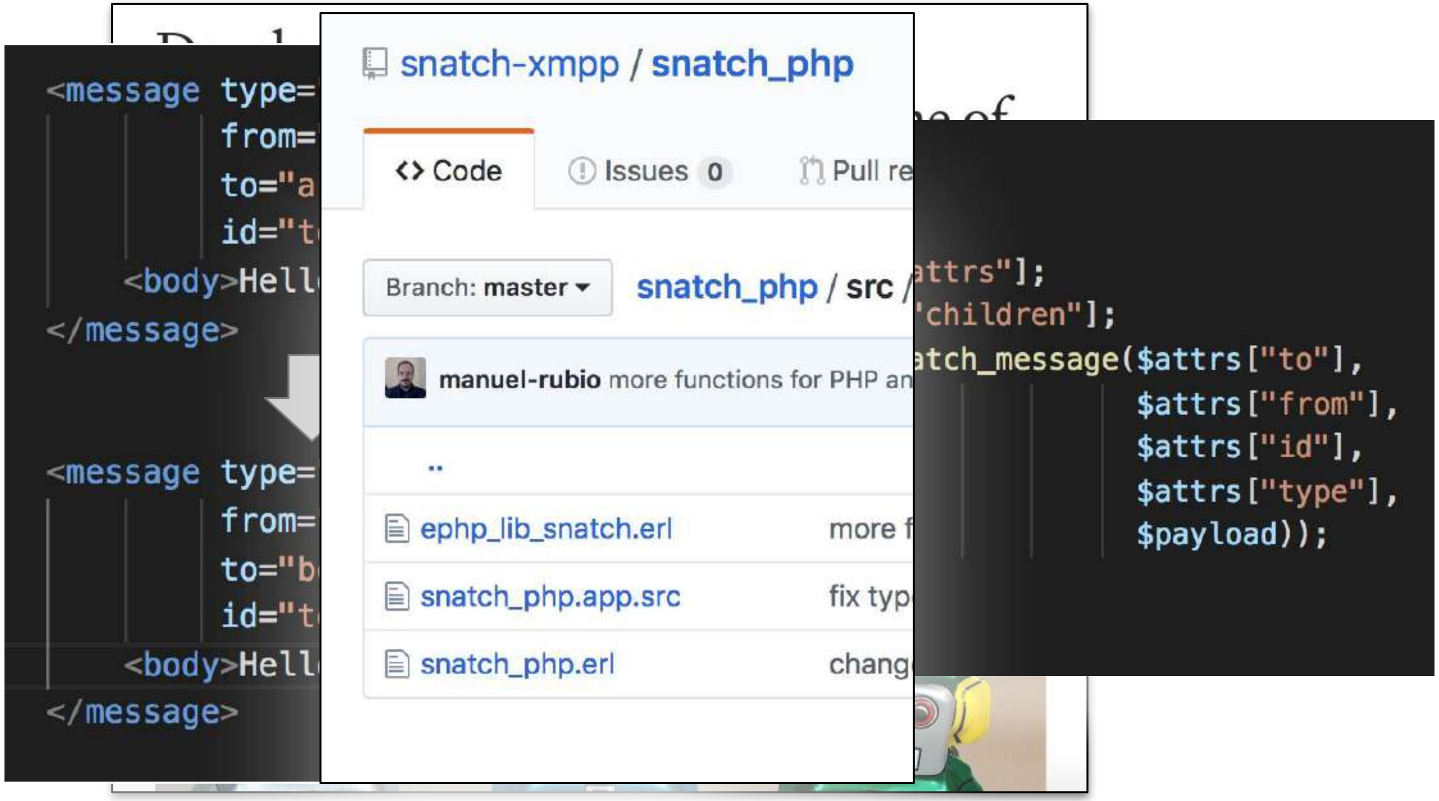
TDD on
single line of

```
<message type="chat"
         from="bob@localhost/pc"
         to="alice@localhost"
         id="test_bot">
    <body>Hello world!</body>
</message>
```

```
<message type="chat"
         from="alice@localhost"
         to="bob@localhost/pc"
         id="test_bot">
    <body>Hello world!</body>
</message>
```

```
'name"]) {

$_REQUEST["attrs"];
 $_REQUEST["children"];
_binary(snatch_message($attrs["to"],
                       $attrs["from"],
                       $attrs["id"],
                       $attrs["type"],
                       $payload));
```

PHP over ERLANG (ephp)



```
<message type=
         from=
         to="a
         id="t
    <body>Hell
</message>

<message type=
         from=
         to="b
         id="t
    <body>Hell
</message>
```

snatch-xmpp / snatch_php

<> Code | Issues 0 | Pull re

Branch: master ▾ snatch_php / src /

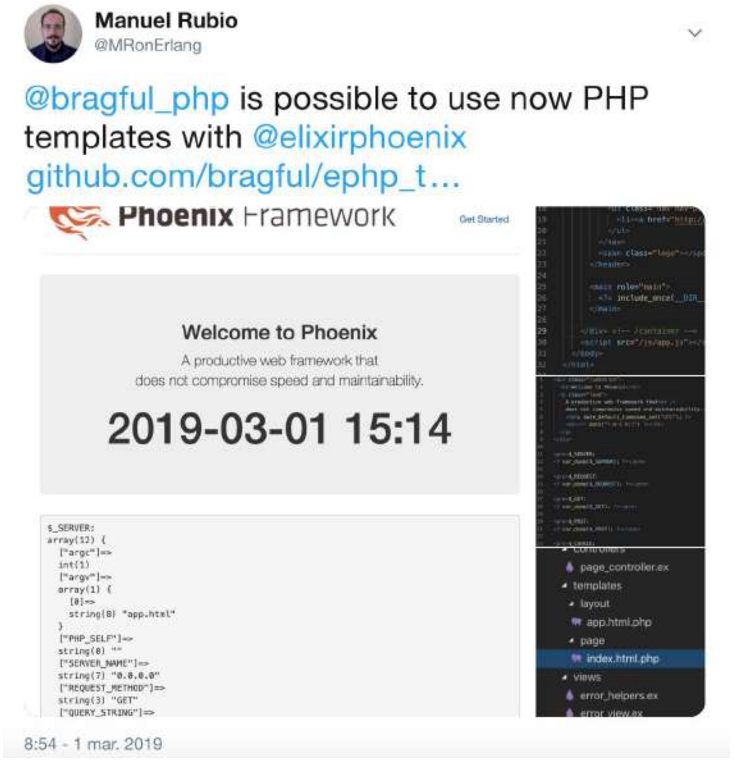
manuel-rubio more functions for PHP an

..

ephp_lib_snatch.erl	more f
snatch_php.app.src	fix typ
snatch_php.erl	chang

```
attrs"];
'children"];
atch_message($attrs["to"],
             $attrs["from"],
             $attrs["id"],
             $attrs["type"],
             $payload));
```

PHP over ERLANG (ephp)



Manuel Rubio
@MRonErlang

@bragful_php is possible to use now PHP templates with @elixirphoenix github.com/bragful/ephp_t…

Welcome to Phoenix

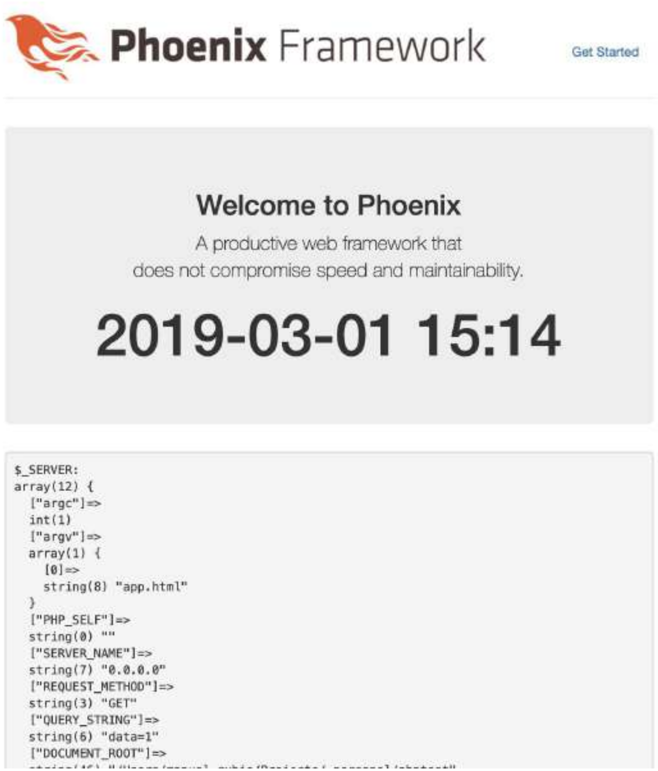
A productive web framework that
does not compromise speed and maintainability.

2019-03-01 15:14

```
$_SERVER:
array(12) {
  ["argc"]=>
  int(1)
  ["argv"]=>
  array(1) {
    [0]=>
    string(8) "app.html"
  }
  ["PHP_SELF"]=>
  string(0) ""
  ["SERVER_NAME"]=>
  string(7) "0.0.0.0"
  ["REQUEST_METHOD"]=>
  string(3) "GET"
  ["QUERY_STRING"]=>
```

8:54 - 1 mar. 2019

PHP over ERLANG (ephp)


Phoenix Framework

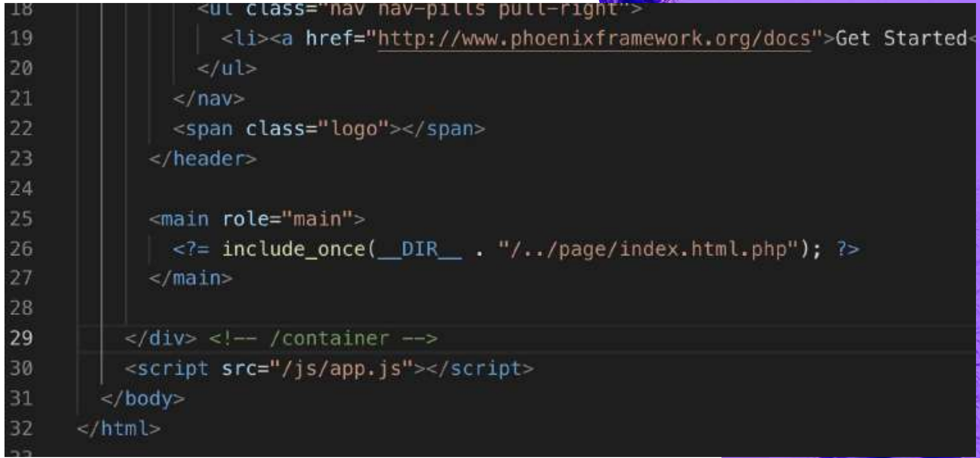
Get Started

Welcome to Phoenix

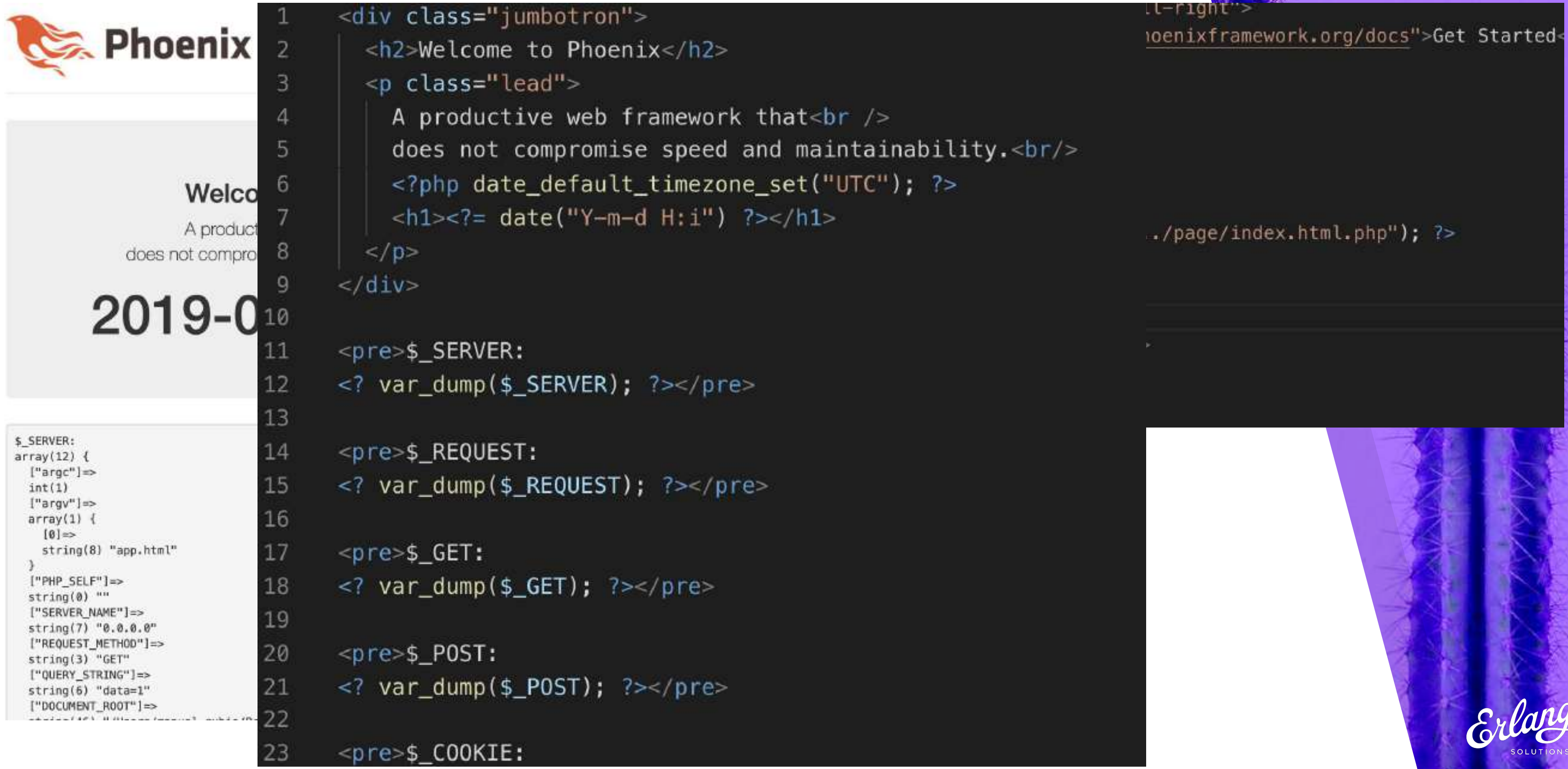
A productive web framework that
does not compromise speed and maintainability.

2019-03-01 15:14

```
$_SERVER:
array(12) {
  ["argc"]=>
  int(1)
  ["argv"]=>
  array(1) {
    [0]=>
    string(8) "app.html"
  }
  ["PHP_SELF"]=>
  string(0) ""
  ["SERVER_NAME"]=>
  string(7) "0.0.0.0"
  ["REQUEST_METHOD"]=>
  string(3) "GET"
  ["QUERY_STRING"]=>
  string(6) "data=1"
  ["DOCUMENT_ROOT"]=>
```


```
          <ul class="nav nav-pills pull-right">
            <li><a href="http://www.phoenixframework.org/docs">Get Started
          </ul>
        </nav>
        <span class="logo"></span>
      </header>

      <main role="main">
        <?= include_once(__DIR__ . "/../page/index.html.php"); ?>
      </main>

    </div> <!-- /container -->
    <script src="/js/app.js"></script>
  </body>
</html>
```

PHP over ERLANG (ephp)



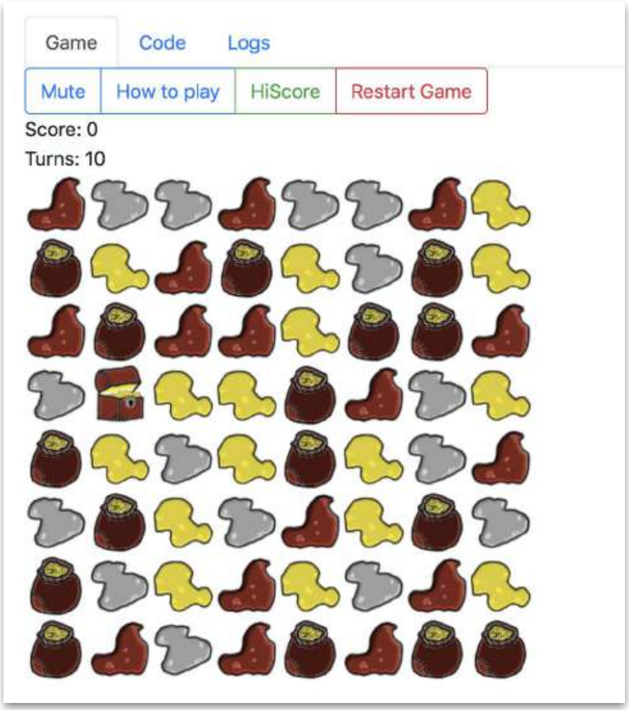
```
$_SERVER:
array(12) {
  ["argc"]=>
  int(1)
  ["argv"]=>
  array(1) {
    [0]=>
    string(8) "app.html"
  }
  ["PHP_SELF"]=>
  string(0) ""
  ["SERVER_NAME"]=>
  string(7) "0.0.0.0"
  ["REQUEST_METHOD"]=>
  string(3) "GET"
  ["QUERY_STRING"]=>
  string(6) "data=1"
  ["DOCUMENT_ROOT"]=>
```

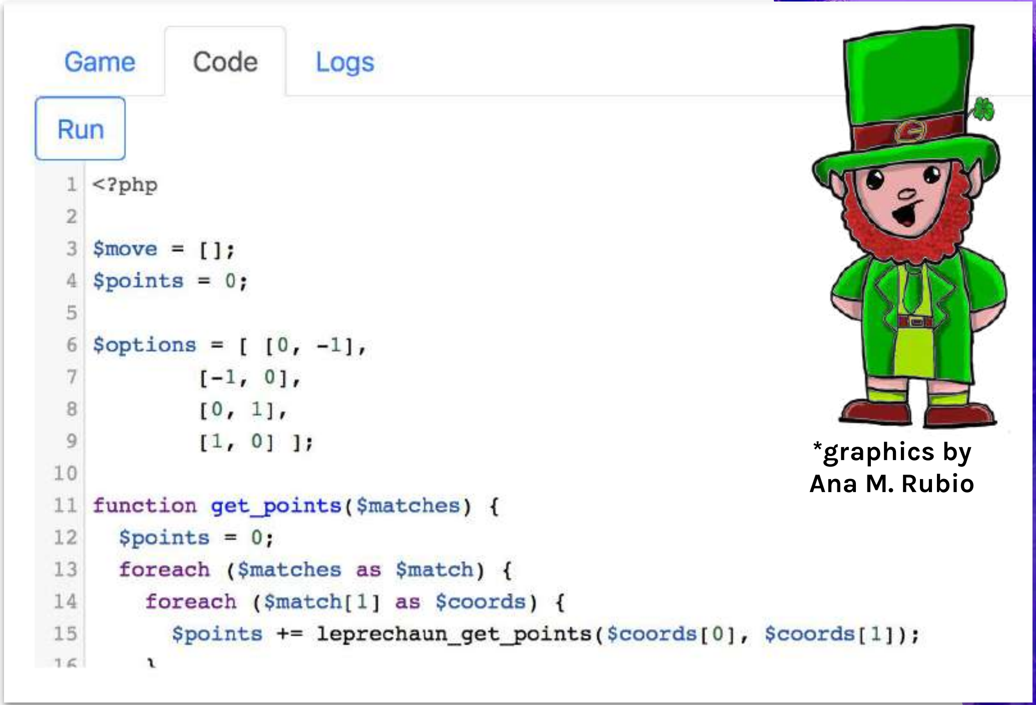
```
<div class="jumbotron">
  <h2>Welcome to Phoenix</h2>
  <p class="lead">
    A productive web framework that<br />
    does not compromise speed and maintainability.<br/>
    <?php date_default_timezone_set("UTC"); ?>
    <h1><?= date("Y-m-d H:i") ?></h1>
  </p>
</div>

<pre>$_SERVER:
<? var_dump($_SERVER); ?></pre>

<pre>$_REQUEST:
<? var_dump($_REQUEST); ?></pre>

<pre>$_GET:
<? var_dump($_GET); ?></pre>

<pre>$_POST:
<? var_dump($_POST); ?></pre>

<pre>$_COOKIE:
```

```
l-right">
oenixframework.org/docs">Get Started

./page/index.html.php"); ?>
```

PHP over ERLANG (ephp)

https://leprechaun.altenwald.com/bot

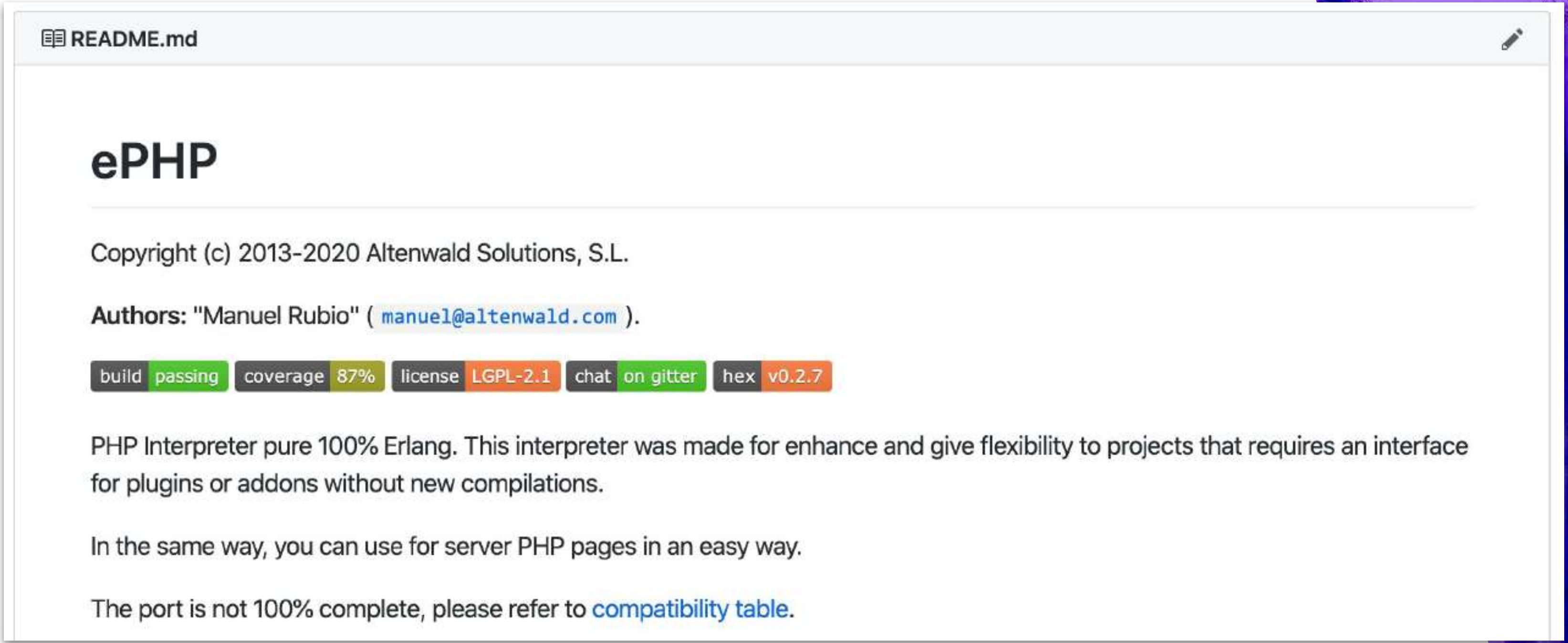




```
<?php

$move = [];
$points = 0;

$options = [ [0, -1],
             [-1, 0],
             [0, 1],
             [1, 0] ];

function get_points($matches) {
  $points = 0;
  foreach ($matches as $match) {
    foreach ($match[1] as $coords) {
      $points += leprechaun_get_points($coords[0], $coords[1]);
```

*graphics by
Ana M. Rubio

PHP over ERLANG (ephp)



README.md

ePHP

Copyright (c) 2013-2020 Altenwald Solutions, S.L.

Authors: "Manuel Rubio" (`manuel@altenwald.com`).

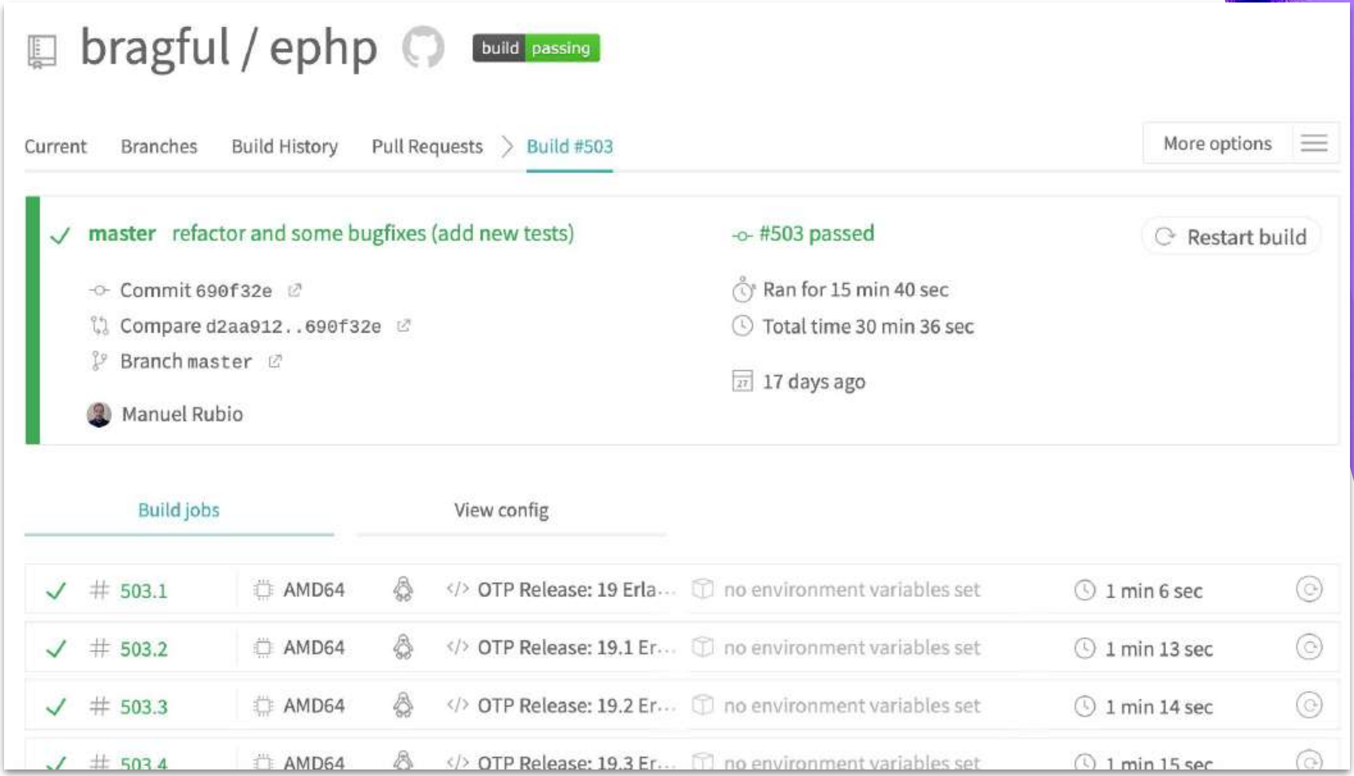
build passing | coverage 87% | license LGPL-2.1 | chat on gitter | hex v0.2.7

PHP Interpreter pure 100% Erlang. This interpreter was made for enhance and give flexibility to projects that requires an interface for plugins or addons without new compilations.

In the same way, you can use for server PHP pages in an easy way.

The port is not 100% complete, please refer to compatibility table.

PHP over ERLANG (ephp)



bragful / ephp build passing

Current | Branches | Build History | Pull Requests > Build #503 | More options

✓ master refactor and some bugfixes (add new tests)

Commit 690f32e
Compare d2aa912..690f32e
Branch master
Manuel Rubio

#503 passed
Ran for 15 min 40 sec
Total time 30 min 36 sec
17 days ago

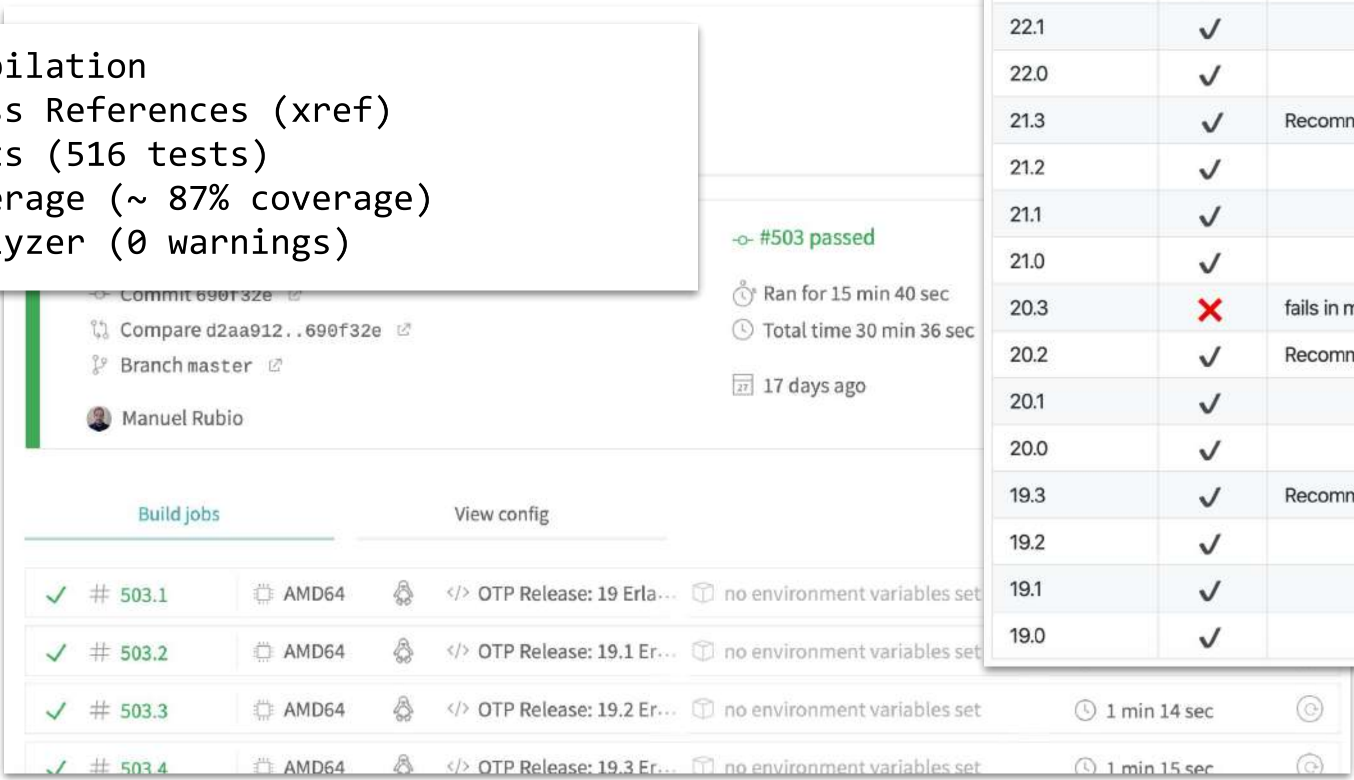
Restart build

Build jobs | View config

✓	# 503.1	AMD64	OTP Release: 19 Erla...	no environment variables set	1 min 6 sec
✓	# 503.2	AMD64	OTP Release: 19.1 Er...	no environment variables set	1 min 13 sec
✓	# 503.3	AMD64	OTP Release: 19.2 Er...	no environment variables set	1 min 14 sec
✓	# 503.4	AMD64	OTP Release: 19.3 Er...	no environment variables set	1 min 15 sec

PHP over ERLANG (ephp)

1. Compilation
2. Cross References (xref)
3. Tests (516 tests)
4. Coverage (~ 87% coverage)
5. Dialyzer (0 warnings)



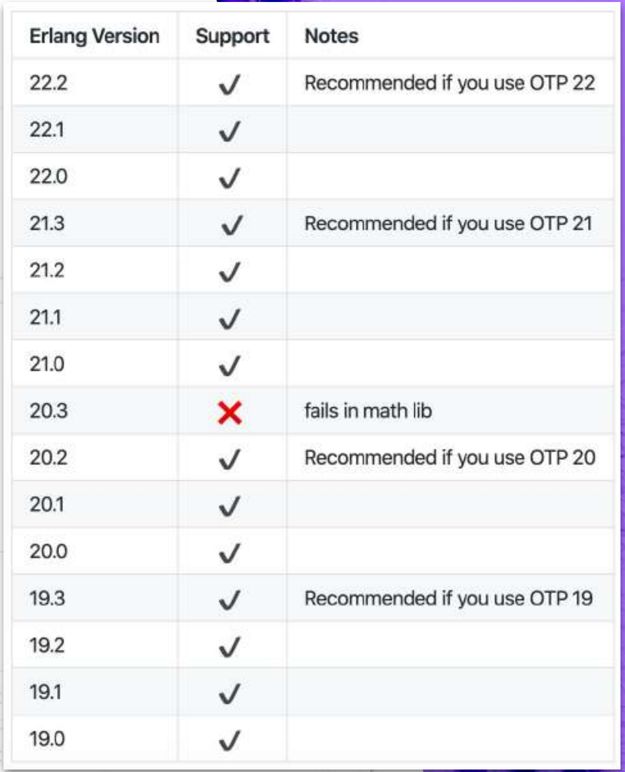
#503 passed

Commit 690f32e
Compare d2aa912..690f32e
Branch master
Manuel Rubio

Ran for 15 min 40 sec
Total time 30 min 36 sec
17 days ago

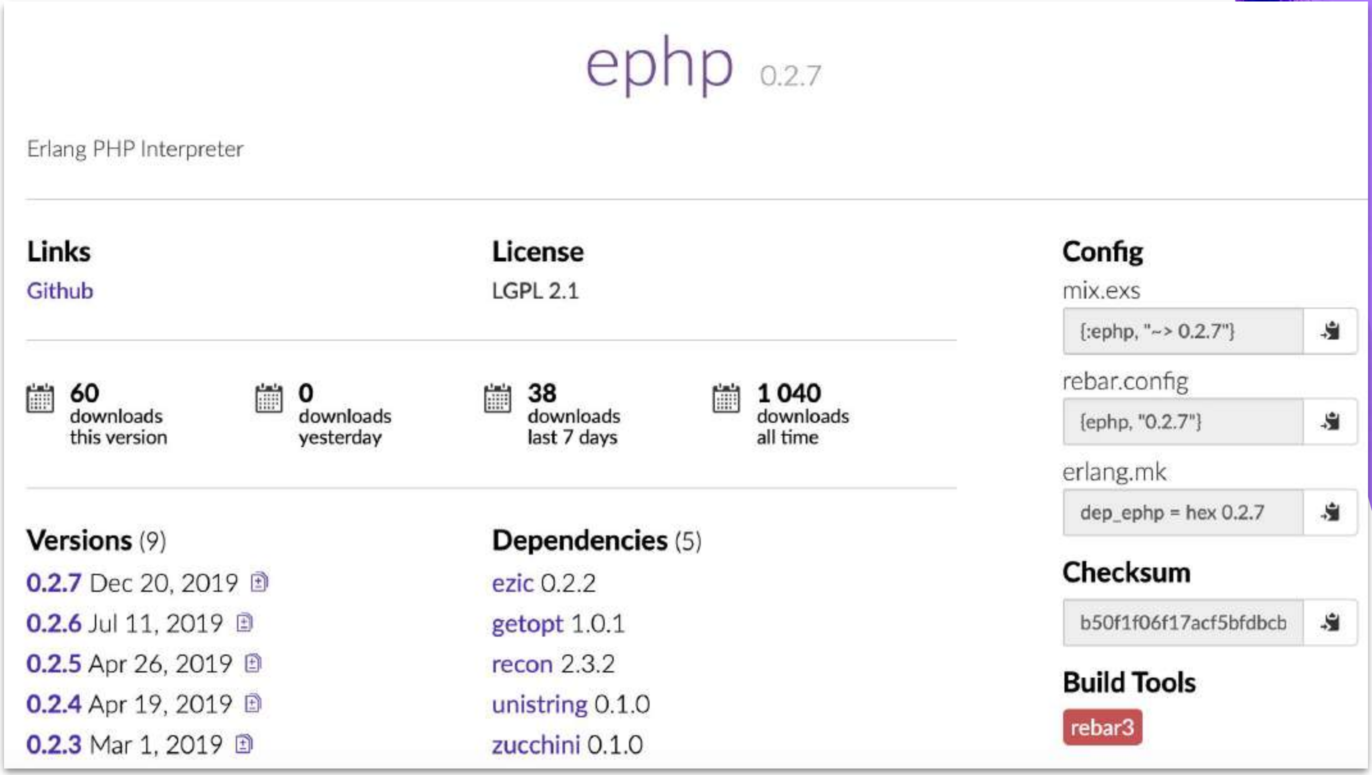
Build jobs View config

✓	# 503.1	AMD64	OTP Release: 19 Erla…	no environment variables set	
✓	# 503.2	AMD64	OTP Release: 19.1 Er…	no environment variables set	
✓	# 503.3	AMD64	OTP Release: 19.2 Er…	no environment variables set	1 min 14 sec
✓	# 503.4	AMD64	OTP Release: 19.3 Er…	no environment variables set	1 min 15 sec



Erlang Version	Support	Notes
22.2	✓	Recommended if you use OTP 22
22.1	✓	
22.0	✓	
21.3	✓	Recommended if you use OTP 21
21.2	✓	
21.1	✓	
21.0	✓	
20.3	✗	fails in math lib
20.2	✓	Recommended if you use OTP 20
20.1	✓	
20.0	✓	
19.3	✓	Recommended if you use OTP 19
19.2	✓	
19.1	✓	
19.0	✓	

PHP over ERLANG (ephp)



ephp 0.2.7

Erlang PHP Interpreter

Links
Github

License
LGPL 2.1

60 downloads this version

0 downloads yesterday

38 downloads last 7 days

1 040 downloads all time

Versions (9)
- 0.2.7 Dec 20, 2019
- 0.2.6 Jul 11, 2019
- 0.2.5 Apr 26, 2019
- 0.2.4 Apr 19, 2019
- 0.2.3 Mar 1, 2019

Dependencies (5)
- ezic 0.2.2
- getopt 1.0.1
- recon 2.3.2
- unistring 0.1.0
- zucchini 0.1.0

Config

mix.exs
```
{:ephp, "~> 0.2.7"}
```

rebar.config
```
{ephp, "0.2.7"}
```

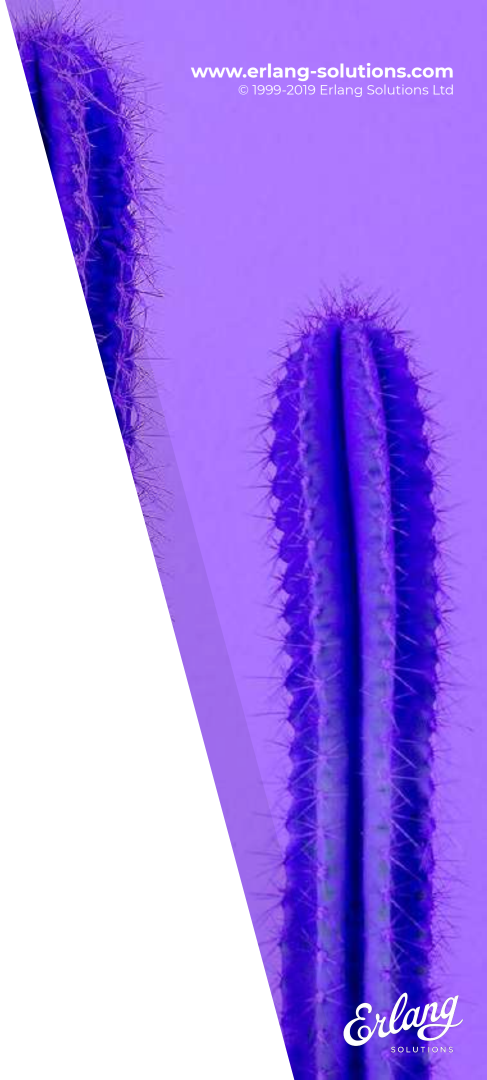
erlang.mk
```
dep_ephp = hex 0.2.7
```

Checksum
```
b50f1f06f17acf5bfdbcb
```

Build Tools
rebar3

Help us to Help you!

- Use it!
- Report bugs!
- Add more functions to **ephp lib modules** and send us your PR!
- Join us to **gitter**!
- Give us **stars in Github**!



THANK YOU
Q&A

Manuel Rubio
Tech Lead Sweden @ Erlang Solutions

@MRonErlang
manuel.rubio@erlang-solutions.com