PHP funcional além do array_map

Jean Carlo Machado

Sobre

CompuFácil

⇢ php-src

⇢ git/git

⇢ torvald/linux

⇢ vim/vim

⇢ Doctrine

⇢ Zend Framework

⇢ phpunit


CODE WRITTEN IN HASKELL IS GUARANTEED TO HAVE NO SIDE EFFECTS.
...BECAUSE NO ONE WILL EVER RUN IT?

Haskell

História

1936 Imperativo

⇢ Alan Turing

⇢ computação universal

⇢ Receita

⇢ assembly

⇢ Goto

⇢ side effects

1950 Estruturado

⇢ Agol

⇢ while, for, if

⇢ reuso

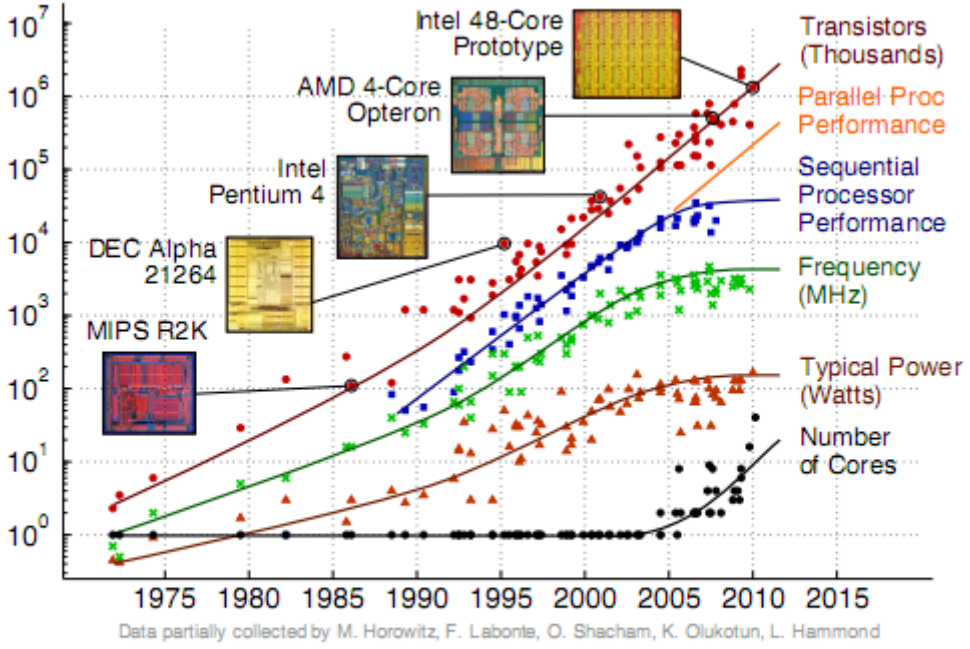
⇢ ~~goto~~

Programação Funcional

~~**mudança de estado**~~

Vantagens

- ⇢ Concorrência nativa
- ⇢ Facilita a modularização
- ⇢ Desacelera o apodrecimento



Prepared by C. Batten - School of Electrical and Computer Engineering - Cornell University - 2005 - retrieved Dec 12 2012 - http://www.csl.cornell.edu/courses/ece5950/handouts/ece5950-overview.pdf

Prática

Impureza === Estado

```
$inpureIncrement = function (&$count) {
    $count++;
};
```

Pureza (relativa)

```
function pureIncrement($count) {
    $myCounter = $count;
    $inpureIncrement = function(&$myCounter){
        $count++;
    };
    $inpureIncrement($myCounter);
    return $myCounter;
}
```

Transparência referencial

```
let y = f x in y + y
// substituível
 f x + f x
```

Impuros por natureza

⇢ IO

⇢ print

⇢ random

Funções de alta ordem

Retornáveis

```php
$sumPartial = function(int $a) {
    return function(int $b) use ($a) {
        return $a+b;
    }
};
$sumOne = $sumPartial(1);
$sumOne(7);
//8
$sumOne(5);
//6
```

Repassáveis

```
$incrementTwo = function ($n, $sumOne) {
    return $sumOne($sumOne($n));
};
```

Fold

Defina o básico

```
function sum($a, $b) {
    return $a + $b;
}
//function product($a, $b);
//function aORb($a, $b);
//function append($a, $b);
//function greater($a, $b);
```

```
function fold(
    callable $f,
    $init,
    $list) {

    if (empty($list)) {
        return $init;
    }
    return $f(
        fold($f, $init, allbutlast($list)),
        last($list)
    );
};
```

Componha

```
$sumList = fold('sum', 0);
$sumList([1,2,3]);
//6
$multiplyList = fold('product', 0);
$sumList([2,2,2]);
//8
```

Map

```
$double = partial('product')(2);
$map($double, [1, 2, 4]));
//[2,4,8]
```

```
$map = function($f, $list) {
    $applyAndAppend=function($f,$list,$b){
        return append($list, $f($b));
    };
    return fold($applyAndAppend,null)($list);
}
```

Árvores

```
$tree = [
    1 => [
        2,
        3 => [
            4,
        ],
    ]
];
foldTree($sum, $sum, 0, $tree)
//10
```

Map Tree

```
$tree = [
    3 => [1,4],
    1 => [1,5 => [1,2,3]]
];
$result  = mapTree($double, $tree);
//[
//    6 => [2,8],
//    2 => [2,10 => [2,4,6]]
//];
```

PHP functions

- ⇢ array_map
- ⇢ array_filter
- ⇢ array_reduce
- ⇢ array_sum
- ⇢ array_unique

g (f input)

⇢ começa f só quando g tenta ler algo

⇢ suspende f

⇢ g roda até precisar ler mais dados

Nth primes

```
function isPrime($x) {
    $seq=\f\takeFrom(
        \f\infiniteSequence(2),
        ($x-2)
    );
    $multipleOfA=
        \f\partial('f\op\aMultipleOfb')
            ($x);
    $result = \f\map($multipleOfA, $seq);
    return \f\noneTrue($result);
}
```

Nth primes

```
function primes() {
    foreach (\f\infiniteSequence(1) as $i) {
        if (isPrime($i)) {
            yield $i;
        }
    }
}

print_r(\f\takefrom(primes(), 10));
//[1, 3, 5, 7, 11, 13, 17, 19, 23, 29]
```

Aplicação parcial

```
$append3 = function($a, $b, $c) {
    return [$a, $b, $c];
};
$append1And2 = \f\partial($append3)(1)(2);
$append1And2(5)
//[1,2,5]
$append1And2(9)
//[1,2,9]
```

Real: get-profile

```
class Database {
    public function query($query, $id) {
        echo "DATABASE_CALL";
        $data = [
            ['id' => 666, 'name' => 'Gandalf'
            ['id' => 777, 'name' => 'Saruman'
        ];
        return array_filter($data,
            function($candidate) use ($id) {
                return $candidate['id'] == $id
            }
        );
    }
}
```

```
function getProfile($database, $userId) {
    return $database->query("
        Select * from User where id = %id",
        $userId
    );
}
$getProfileConcrete = \f\partial('getProfile')
```

```
$memoize = function($func) {
    static $results = [];
    return function ($a) use ($func, &$results
        $key = serialize($a);
        if (empty($results[$key])) {
            $results[$key] = call_user_func($f
            return $results[$key];
        }

        return $results[$key];
    };
};
$memoizedProfile = $memoize($getProfileConcret
```

```
$logger = function($str) {
    echo $str;
};

function logService($logger, $serviceName, cal
    $logger("Service called ".$serviceName." w
    return call_user_func($service, $arg);
};

$loggedGetProfile = \f\partial('logService')($
```

```
//DATABASE_CALL
//Array
//(
//    [0] => Array
//        (
//            [id] => 666
//            [name] => Gandalf
//        )
//
//)
//Service called getProfile with params i:666;
//(
//    [0] => Array
//        (
//            [id] => 666
//            [name] => Gandalf
```

Modularização

- Mais que módulos
- Decompor os problemas em partes menores
- Re-compor com avaliação tardia e funções de alta orem

~~Apodrecimento~~

- ⇢ Impossível adicionar efeitos sem quebrar as interfaces
  - ⇢ Quão maior a interface mais feio o código
- ⇢ Interfaces facilmente quebráveis com composicão
  - ⇢ Quebrar encoraja reúso
- ⇢ Complexidade horizontal ao invés de vertical

Conclusão

⇢ Imperativo quando necessário

⇢ Funcional quando possível

⇢ Para novos tipos:

⇢ defina as operações fundamentais

⇢ junte funções

⇢ Seu trabalho não é resolver problemas

⇢ Definir problemas de uma forma que eles se resolvam

Ferramentas

- hlstrojny/functional-php
- functional-php/pattern-matching
- jeanCarloMachado/f
- pimple/pimple

Referências

⇢ Why functional programming matters

⇢ http://archive.li/uTYWZ#selection-407.912-407.919

⇢ https://medium.com/@jugoncalves/functional-programming-should-be-your-1-priority-for-2015-47dd4641d6b9

⇢ https://blog.inf.ed.ac.uk/sapm/2014/03/06/enemy-of-the-state-or-why-oop-is-not-suited-to-large-scale-software/

⇢ https://www.youtube.com/watch?v=7Zlp9rKHGD4

⇢ https://www.youtube.com/watch?v=q0HRCEKAcas

Dúvidas?

Github: https://github.com/jeanCarloMachado

Twitter: https://twitter.com/JeanCarloMachad

E-mail: contato@jeancarlomachado.com.br