# Como testar unitariamente em PHP

Tales Mota Machado[1][2]

[1]Universidade Federal de Ouro Preto

[2]Analista de Sistemas -GerenciaNet-

Uma visão geral sobre teste unitários.

# Outline

O que é qualidade?

# O que é qualidade?

QUALIDADE

O dicionário Michaelis define qualidade como: “2 Excelência, virtude, talento.”, “4 Grau de perfeição, de precisão, de conformidade a um certo padrão.”

# O que é um código de qualidade?

"Qualidade depende do ponto de vista!"[2]

- Do usuário: satisfação de requisitos, ausência de bugs, frequência de releases, bom desempenho, etc.
- Do desenvolvedor: facilidade de entender o código, flexibilidade para modificar o código, abrangência da suíte de testes, etc.

# Ausência de bugs

É possível escrever um código livre de bug?

# o que é um bug?

O que é um bug?

# Impossível

É impossível[1]. Por mais simples que seja o código, não há como garantir que não haverá bugs devido às várias dependências:

- Compilador/interpretador
- SO
- Usuário final

Toda via o correto seria dizer minimizar a ocorrência de bugs.

Existem vários framework's para automatizar os testes de unidade. Os mais usados no mercado sem dúvida são os xUnit, abrangindo a quase todas linguagens. Para PHP temos:

- PHPUnit
- **SimpleTest**

# SimpleTest

simpletest “http://www.simpletest.org/” usa a mesma API do xUnit:

| | |
|---|---|
| assertTrue ($ x) | Falha se $ x é falsa |
| assertFalse ($ x) | Falha se $ x é verdadeiro |
| assertNull ($ x) | Falha se $ x é definido |
| assertNotNull ($ x) | Falhar se não definir $ x |
| assertIsA ($ x, $ t) | Falha se $ x não é o tipo ou classe $ t |
| assertNotA ($ x, $ t) | Falha se $ x é da classe ou tipo de $ t |
| assertEqual ($ x, $ y) | Falha se $ x == $ y é falso |
| assertNotEqual ($ x, $ y) | Falha se $ x == $ y é verdadeiro |
| assertWithinMargin ($ x, $ y, $ m) | Falhar se abs ($ x - $ y) <$ m é falsa |

| | |
|---|---|
| assertOutsideMargin ($ x, $ y, $ m) | Falhar se abs ($ x - $ y) <$ m é verdade |
| assertIdentical ($ x, $ y) | Falhar se y $ x == $ é falsa ou uma incompatibilidade de tipo |
| assertNotIdentical ($ x, $ y) | Falhar se y $ x == $ é verdadeiro e tipos de correspondência |
| assertReference ($ x, $ y) | Falhar a menos que $ e $ x y são a mesma variável |
| assertClone ($ x, $ y) | Falhar a menos que $ e $ x y são cópias idênticas |
| assertPattern ($ p, $ x) | Falhar a menos que a regex $ p $ x corresponde |
| assertNoPattern ($ p, $ x) | Falhar se a regex $ p $ x corresponde |

| | |
|---|---|
| expectError ($ x) | Falhar se erro de correspondência não occour |
| expectException ($ x) | Falha se a exceção de correspondência não é lançada |
| ignoreException ($ x) | Engole qualquer exceção próximo correspondência |
| assert ($ e) | Falha na falhou expectativa objeto $ e |

# Mock

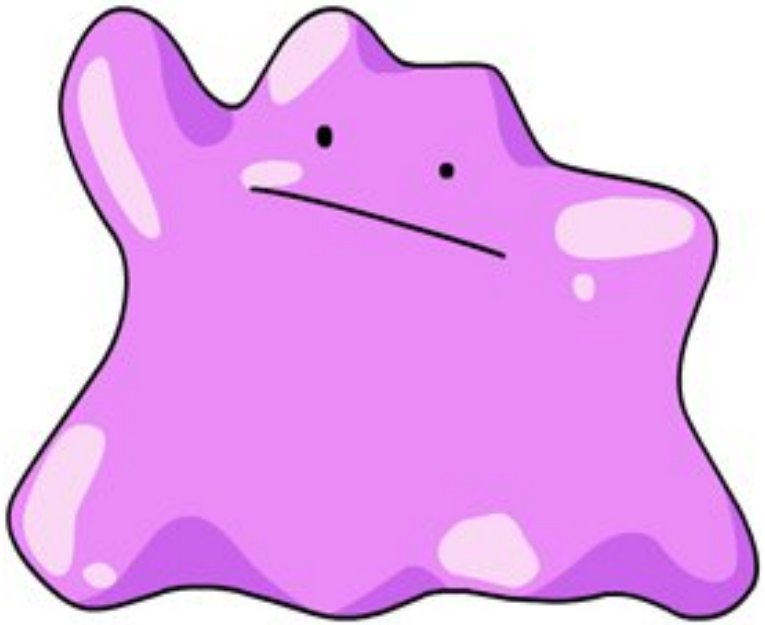

**Figure:** Mock1

# Mock

**Figure:** Mock2

 *Engenharia de software* Ian Sommerville , 2011, 9ª Edição

 *Qualidade de Código: mantendo seu projeto de software sob controle* 2009, Antonio Terceiro , *"http://softwarelivre.org/terceiro/blog/qualidade-de-codigo-mantendo-seu-projeto-de-software-sob-controle"*

 *Prática: melhore a qualidade do código para evitar uma enchente de bugs* 2010, Guilherme Silveira, Caelum , *"http://blog.caelum.com.br/pratica-melhore-a-qualidade-do-codigo-para-evitar-uma-enchente-de-bugs/"*