パラメタライズドテスト

2022/07/08 株式会社ラグザイア 中島 滋

パラメタライズドテストとは？

テストコードの書き方
名前がかっこいい

FizzBuzz

1から100までの数に対して
3で割り切れる数はFIZZ
5で割り切れる数はBUZZ
3でも5でも割り切れる数はFIZZBUZZと表示する。
それ以外の数は数字のまま表示する。

どんなテストをかきますか？

まず

3、5、15は入力します。

他にはなにかいれますか？

3、5、15のまわりの数字はどうでしょうか？

2，4，6，14，16 …

1から100までの数 以外はどうなるでしょう？
0、1と100、101 も確認したいです。


RSpec

```
context '数が1より小さいとき' do
  it { expect(fizzbuzz 0).to eq '' }
end

context '数が100より大きいとき' do
  it { expect(fizzbuzz 101).to eq '' }
end

context '数が3で割り切れるとき' do
  it { expect(fizzbuzz 3).to eq 'FIZZ' }
  it { expect(fizzbuzz 6).to eq 'FIZZ' }
  it { expect(fizzbuzz 9).to eq 'FIZZ' }
end
```

```
context '数が5で割り切れるとき' do
  it { expect(fizzbuzz 5).to eq 'BUZZ' }
  it { expect(fizzbuzz 10).to eq 'BUZZ' }
  it { expect(fizzbuzz 20).to eq 'BUZZ' }
end

context '数が15で割り切れるとき' do
  it { expect(fizzbuzz 15).to eq 'FIZZBUZZ' }
  it { expect(fizzbuzz 30).to eq 'FIZZBUZZ' }
  it { expect(fizzbuzz 45).to eq 'FIZZBUZZ' }
end
```

```
context '数が3でも5でも割り切れないとき' do
  it { expect(fizzbuzz 1).to eq '1' }
  it { expect(fizzbuzz 2).to eq '2' }
  it { expect(fizzbuzz 4).to eq '4' }
  it { expect(fizzbuzz 7).to eq '7' }
  it { expect(fizzbuzz 8).to eq '8' }
  it { expect(fizzbuzz 11).to eq '11' }
  it { expect(fizzbuzz 13).to eq '13' }
  it { expect(fizzbuzz 14).to eq '14' }
  it { expect(fizzbuzz 16).to eq '16' }
end
```

```
expect(fizzbuzz 3).to eq 'FIZZ'
```

が、一杯でてくるのが気になる。

Don't repeat yourself

アサーション（expect~）の重複を解消する
パラメタライズドテスト

```
[
  [0, ''],
  [1, '1'],
  [2, '2'],
  [3, 'FIZZ'],
  [4, '4'],
  [5, 'BUZZ'],
  # 中略
  [101, '']
].each do | number, answer |
  it { expect(fizzbuzz number).to eq answer }
end
```

配列で、入力値と期待する結果をまとめる

```
expect(fizzbuzz number).to eq answer
```

をひとつに

パラメタライズドテストを

サポートしている

テスティングフレームワーク


NUnit

```
[TestCase(0, ExpectedResult = "")]
[TestCase(1, ExpectedResult = "1")]
[TestCase(2, ExpectedResult = "2")]
[TestCase(3, ExpectedResult = "FIZZ")]
[TestCase(4, ExpectedResult = "4")]
[TestCase(5, ExpectedResult = "BUZZ")]
// 中略
[TestCase(14, ExpectedResult = "14")]
[TestCase(15, ExpectedResult = "FIZZBUZZ")]
[TestCase(16, ExpectedResult = "16")]
[TestCase(100, ExpectedResult = "BUZZ")]
[TestCase(101, ExpectedResult = "")]
public string FizzBuzzTest(int number)
{
    return FizzBuzz(number);
}
```

属性でパラメーターと期待する値を指定
テストケースはひとつ

自分でeachやブロックを書かなくてよい
誰が書いても同じ書き方になる
テストコードが読みやすい

.NET の世界のから

自慢 情報共有