




Fscheck — альтернативный путь для unit-тестов?

Dodo Pizza, Юлия Ковалева
ноябрь 2017 года

Обо мне



 juliana_kov

 yuka.julia

- Первая встреча с тестами случилась 5 лет назад.
- Автоматизирую и оптимизирую тесты последние 3 года.
- Увлеченно борюсь за эффективность проверок.

Я проверил(-а) свой код

Я проверил(-а) свой код

**Я написал(-а) шаблон unit
теста**

Я проверил(-а) свой код

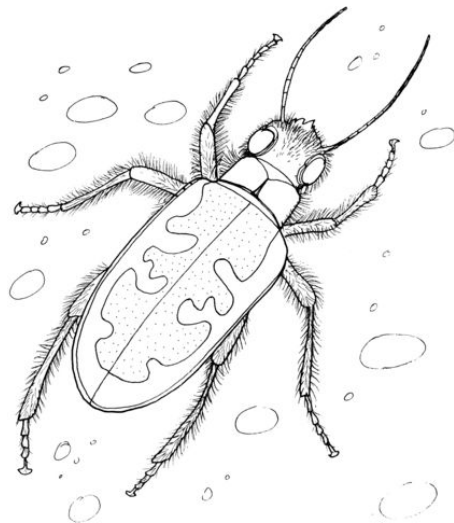
Я написал(-а) шаблон unit
теста

**Я выбрал(-а) парочку
примеров для теста**



Откуда взялись баги?

QA противостоит багам



**Всегда ли вы пишете unit-
тесты?**

Наша основная боль

<http://dodois.com>

MSK
DOT
NET

Ручной регресс

- Автотесты UI: 30
- Unit-тесты: низкий процент покрытия тестами кода (≈ 2330)
- Продолжительность регресса: 2-3 дня

Требование



Dodo IS / DODOIS-15969

Функция для проверки валидности периода в рамках одного дня

Реализовать функцию, которая должна вернуть TRUE, если период валидный и FALSE в противном случае.

Валидным считается период, когда Начало Периода < Окончания Периода.

09:00-08:00 и 09:00-09:00 - невалидные периоды

08:00-09:00 и 08:01-08:05 - валидные периоды

Unit-Test

```
[Test]
[TestCase("09:00-08:00")]
[TestCase("09:00-09:00")]
public void EndTimeShouldBeGreaterThanBeginTime(string period)
{
    var timePeriod = period.TimePeriod();

    var result = CellValidator.IsPeriodValid(timePeriod);

    Assert.IsFalse(result);
}
```

Есть ли проблемы?

Unit-Test

```
[Test]
[TestCase("09:00-08:00")]
[TestCase("09:00-09:00")]
public void EndTimeShouldBeGreaterThanBeginTime(string period)
{
    var timePeriod = period.TimePeriod();

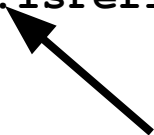
    var result = CellValidator.IsPeriodValid(timePeriod);

    Assert.IsFalse(result);
}
```

```
public void EndTimeShouldBeGreaterThanBeginTime(string period)
{
    var timePeriod = period.TimePeriod();

    var result = CellValidator.IsPeriodValid(timePeriod);

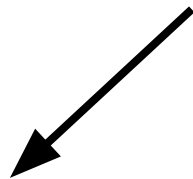
    Assert.IsFalse(result);
}
```



timePeriod на самом деле совсем не time,
так как 24:61-25:62 - это валидный период

Unit-тест

```
[Test]  
[TestCase("09:00-08:00")]  
[TestCase("09:00-09:00")]
```



крайние значения для 12
часового формата
00:00-00:00
11:59-11:58

крайние значения для 24
часового формата
23:59-23:54

Больше кейсов

```
[Test]
```

```
[TestCase("09:00-08:00")]
```

```
[TestCase("09:00-09:00")]
```



```
[Test]
```

```
[TestCase("08:00-08:01")]
```

```
[TestCase("09:00-10:00")]
```

```
[TestCase("00:00-12:00")]
```

```
[TestCase("23:58-23:59")]
```

```
[TestCase("11:59-23:59")]
```



```
[Test]
```

```
[TestCase("23:59-00:00")]
```

```
[TestCase("24:00-24:05")]
```

```
[TestCase("09:00-08:00")]
```

```
[TestCase("09:00-09:00")]
```


Ещё больше кейсов

```
[Test]
[TestCase("09:00-08:00")]
[TestCase("09:00-09:00")]
```



```
[Test]
public void PeriodIsValid(
    [Random(0,23,10)] int h,
    [Random(0,59,10)] int min){
    ...
}
```

Без техник Test Design-а никак

Pairwise - каждое тестируемое значение каждого проверяемого параметра хотя бы 1 раз сочетается с каждым значением остальных параметров.

```
[Test]
[TestCase("09:00-08:00")]
[TestCase("09:00-09:00")]
```



```
[Test, Pairwise]
public void PeriodIsValid(
    [Values("00", "12", "23")] string h1,
    [Values("00", "58", "10")] string
min1,
    [Values("00", "13", "23")] string h2,
    [Values("01", "59")] string min2) {
    ... }
}
```

<https://github.com/nunit/docs/wiki/Pairwise-Attribute>

Тестовые данные отдельно от тестов



[Test]

[TestCase("09:00-08:00")]

[TestCase("09:00-09:00")]

TEST-1	Валиден ли период?		
Кейс Id	Период	Описание	Ожидаемый результат
TMP-1	08:00-09:00	часы начало < часы окончание	истина
TMP-2	09:00-09:01	минуты начало < минуты окончание	истина
TMP-3	00:00-12:00	время начала = времени окончания, если 12-часовой формат	истина
TMP-4	23:58-23:59	крайнее верхнее значение для часов	истина
TMP-5	11:59-12:56	крайнее верхнее значение для минут	истина
TMP-6	08:00-08:00	время начала = времени окончания	ложь

**А если хочется доверять
своему коду?**

John Hughes



Don't Write Tests!

John Hughes

A screenshot of an Erlang shell window titled 'Erlang'. The window shows two test runs. The first run, labeled '37>', shows 'OK, passed 100 tests' followed by a list of statistics: {q,size,1} at 37.0%, {q,put,2} at 33.0%, {q,get,1} at 22.1%, and {q,new,1} at 7.8%. The second run, labeled '38>', also shows 'OK, passed 100 tests' with slightly different statistics: {q,size,1} at 38.4%, {q,put,2} at 32.7%, {q,get,1} at 22.1%, and {q,new,1} at 6.8%. The window has a standard Windows taskbar at the bottom showing the date and time as 15:13 on 2017-06-19.

https://www.youtube.com/watch?v=hXnS_Xjwk2Y

Property Based подход

- TDD, где свойства и классы рассматриваются как правила.
- Правило проверяется 100 раз на сгенерированных автоматически данных, которые удовлетворяют предусловиям.


Примеры правил

ГлавнаяЗаказыСменыКурьерыСтопСырьеПриемГрафикиОбедыCall-центрУстройства

Вельск-1русский


Поиск по продуктам

ВсеПитцыЗакускиНапиткиДесертыСоусыДругое




Двойная пепперони

25T30



Додо

25T30





Дон Бекон


25T30

395 ₽

В корзину







Корзина (0)Промо

Продукты

СоусыДоступно бесплатно: 0

Сгущенное молоко 1 шт.00₽

Соус Барбекю Додо 1 шт.00₽

Соус Ранч Додо 1 шт.00₽

Самовывоз

Продукты:0₽

Сувениры:0₽

Скидка:0₽

Сумма заказа:0₽

Оплата:Заказ от агрегатора

Наличные

По карте в пиццерии

У клиента:

В корзине пусто

Анна (Вельск, 17:13)

Сумма заказа: 0 ₽

17:38

Промокод

Примеры правил

Заказ без выпекаемого продукта после оплаты переходит в статус "Готов".

Корзина покупателя:

Газированная вода 1л
Маффин 1 шт
Морс 1 шт



01:00
1 - 0
Без Имени
15 руб.
ГОТОВ

Какие есть Property Based фреймворки?

Для мира .Net - FsCheck

<https://fscheck.github.io/FsCheck/>

Остальной мир:

- Java - Java QuickCheck
- Scala - ScalaCheck
- JavaScript - QC.js
- Ruby - Rantly
- Есть также фреймворки для Clojure, Python, Groovy

Пользуетесь ли вы FsCheck?

0 votes
A: Customise FsCheck output
 You can also use labels to display whatever you want when a test fails:
<https://fscheck.github.io/FsCheck/Properties.html#And-Or-and-Labels> ...
 answered Aug 18 by Kurt Schelfthout

4 votes
A: Force FsCheck to generate NonEmptyString for discriminating union fields of type string
 This goes against the grain of property based testing (in that you explicitly prevent valid test cases from being generated), but you could wire up the non-empty string generator to be used for all st ...
 answered Aug 15 by scrwtp

7 votes
Q: Force FsCheck to generate NonEmptyString for discriminating union fields of type string
 I'm trying to achieve the following behaviour with **FsCheck**: I'd like to create a generator that will generate a instance of MyUnion type, with every string field being non-null/empty. type ... MyNestedUnion = | X of string | Y of int * string type MyUnion = | A of int * int * string * string | B of MyNestedUnion My 'real' type is much larger/deeper than the MyUnion, and **FsCheck** is ...
 testing # fscheck
 asked Aug 15 by Edward Dewhurst

1 answer

Инструмент	Количество вопросов
FsCheck	263
ScalaCheck	641
JavaQuickCheck	70
QC	5
Haskell	769

Научные статьи, посвященные FsCheck

март 2016 года, статья от представителей Graz University of Technology из Австрии

Property-based Testing with FsCheck by Deriving Properties from Business Rule Models

http://truconf.ist.tugraz.at/wp-content/uploads/2016/03/AMOST_2016_Aichernig_Schumi.pdf

```
[rev,id  
]
```

```
weave' 16 (jux (iter 4) $ smash 4  
-1 speed "2"
```

```
51:4: 9682
```

```
karlsruhe.tidal Top L10
```

```
Data.Maybe| tidal> tidal
```

```
UTC,89.637343,1.1
```

```
tidal> tidal> tidal> tid
```

```
al.VolcaKeys Sound.OSC.F
```

```
Tidal.VolcaKeys Sound.OS
```

```
nd Tidal VolcaKeys Sound
```

Структура простого теста на FsCheck

```
[Test(Description = "Успешное преобразование строки к формату N-N")]  
public void ShouldFormatStringToOrderNumber_Fs2()  
{  
    var values = Gen.Elements("1-1", " 2-2 ", " 321-93", "1 - 1",  
"343fg - rr4d").ToArbitrary();  
    Prop.ForAll(values,  
        val => val.GetFormattedOrderNumberText()  
            .Equals(Regex.Replace(val, "[^0-9-]*", ""))  
    ).QuickCheck();  
}
```

Пример 1 (Dodo IS)

```
[Test]
public void PauseForCellWillBeCreated_IfAfterSellCellIsEmpty()
{
    Prop.ForAll<DateTime>(currentDate =>
    {
        //удалено вычисление cellSettings
        SellPieces(currentDate, cellSettings);
        AssertPauseWasOpenedForUnit(cellSettings, currentDate);
    })
    .QuickCheck();
}
```

С какими параметрами запустился тест?

Пример 2

```
[Test]
public void PauseForCellWillBeCreated_IfAfterSellCellIsEmpty()
{
    Prop.ForAll<DateTime>(currentDate =>
    {
        //удалено вычисление cellSettings
        SellPieces(currentDate, cellSettings);
        AssertPauseWasOpenedForUnit(cellSettings, currentDate);
    })
    .VerboseCheckThrowOnFailure();
}
```

**Сколько параметров можно использовать
для проверки правила?**

Пример 3

```
[Test(Description = "Успешное преобразование строки к формату N-N")]  
public void ShouldFormatStringToOrderNumber_Fs3()  
{  
    var separators = Gen.Elements("-", " - ", "- ").ToArbitrary();  
    var values1 = Gen.Choose(0, 600).ToArbitrary();  
    var values2 = Gen.Choose(0, 600).ToArbitrary();  
    Prop.ForAll(values1, values2, separators,  
        (val1, val2, sep) =>  
        {  
            var text = (val1 + sep + val2);  
            return text.GetFormattedOrderNumberText()  
                .Equals(Regex.Replace(text, "[^0-9-]*", ""));  
        }).VerboseCheckThrowOnFailure();  
}
```

**Если требуется варьировать “кастомный”
параметр ...**

Пример 4 (без FsCheck)

```
[Test]
[TestCase("09:00-08:00")]
[TestCase("09:00-09:00")]
public void EndTimeGreaterThanBeginTime(string period)
{
    var timePeriod = period.TimePeriod();
    var result = CellValidator
        .IsPeriodValid(timePeriod);
    Assert.IsFalse(result);
}
```

Пример 4 (с FsCheck)

```
[Test]
public void EndTimeGreaterThanBeginTime()
{
    Prop.ForAll(GetPeriods(), period =>

        !CellValidator.IsPeriodValid(period.GetPeriod()))
        .VerboseCheckThrowOnFailure();
}
```

Генератор для кастомного типа

```
private Arbitrary<TimePeriod> GetPeriods()  
{  
    var random = new System.Random();  
    var periods = Arb.From<int>()  
        .Convert(i =>  
            {  
                var begin = Math.Abs(i * 10) + random.Next(0, 600);  
                var end = Math.Abs(i * 10);  
                return new TimePeriod(begin, end);  
            }, i => random.Next(0, 1440));  
    return periods;  
}
```

Пример 5 (с FsCheck)

```
[Test(Description = "Успешное преобразование строки к формату N-N")]  
public void ShouldFormatStringToOrderNumber_Fs5()  
{  
    Prop.ForAll(GetNumbers(), number =>  
        {  
            var text = number.ToString();  
            return text.GetFormattedOrderNumberText()  
                .Equals(Regex.Replace(text, "[^0-9-]*", ""));  
        }).VerboseCheckThrowOnFailure();  
}
```


Генератор для кастомного типа

```
private Arbitrary<NumberOfOrder> GetNumbers()  
{  
    string[] seps = { "-", " - ", "- ", " -", "ie- fgd", "#-#" };  
    var random = new Random();  
    var periods = Arb.From<Tuple<int, int>>()  
        .Filter(tuple => tuple.Item1 >= 0 && tuple.Item2 >= 0)  
        .Convert(tuple =>  
            {  
                var begin = tuple.Item1 * 10;  
                var end = tuple.Item2 * 10;  
                var sep = seps[random.Next(0, seps.Length - 1)];  
                return new NumberOfOrder(begin, sep, end);  
            }, tuple => new Tuple<int, int>(0, 0));  
    return periods;  
}
```

100 комбинаций - это много и долго, можно ли урезать?



Пример 6

```
[Test(Description = "Успешное преобразование строки к формату N-N")]  
public void ShouldFormatStringToOrderNumber_Fs1()  
{  
    var conf = new Configuration { MaxNbOfTest = 10 };  
    var values = Gen.Elements("1-1", " 2-2 ",  
                             " 34521-9943", "1 - 1").ToArbitrary();  
    Prop.ForAll(values, val => val.GetFormattedOrderNumberText()  
                  .Equals(val.Replace(" ", ""))  
    ).Check(conf);  
}
```

Функции генерации данных

1. Gen.Constant
 2. Gen.Choose
 3. Gen.Elements
 4. Gen.Map
 5. Gen.ListOfLength
- и другие варианты

<https://fscheck.github.io/FsCheck/TestData.html>

Как можно фильтровать сгенерированные данные для нужд теста?

Gen.Filter (TryFilter) или Gen.Where

Arb.Filter

Пример 7

```
var periods = Arb.From<Tuple<int, int>>()  
    .Filter(tuple => tuple.Item1 >= 0 && tuple.Item2 >= 0)  
    .Convert(tuple =>  
        {  
            var begin = tuple.Item1 * 10;  
            var end = tuple.Item2 * 10;  
            var sep = separators[RandomIndex()];  
            return new NumberOfOrder(begin, sep, end);  
        }, tuple => new Tuple<int, int>(0, 0));
```

Пример 8

```
private Arbitrary<double> GetMinutes()  
{  
    var periods = Arb.From<double>()  
        .MapFilter(i => i.Round(1),  
                   i => i >= 0  
                   && i <= 8  
        );  
    return periods;  
}
```

**Можно ли как-то распределить
генерируемые тестовые данные в группы?**

Пример 9

```
[Test(Description = "Успешное преобразование строки к формату N-N" )]  
public void ShouldFormatStringToOrderNumber_Fs6()  
{  
    Prop.ForAll(GetNumbers(),  
        number =>  
        {  
            var text = number.ToString();  
            var isEqualTexts = text.Equals(ReplaceNotDigitsAndHiphen(text));  
            return text.GetFormattedOrderNumberText()  
                .Equals(ReplaceNotDigitsAndHiphen(text))  
                .Classify(isEqualTexts, "у текста нет лишних символов")  
                .Classify(!isEqualTexts, "текст подлежит изменениям");  
        }).VerboseCheckThrowOnFailure();  
}
```

Зачем же вся эта презентация?

1. Это рассказ о том, чем Property Based Approach отличается от параметризованных тестов.
2. На реальных примерах разобраны возможности FsCheck.

Удачи в экспериментах!
Генерируйте тестовые данные ...

Особое внимание

- Используются для разных целей: Gen и Arb
- Мах. 3 параметра теста могут задаваться генераторами
- Тестовые данные можно разбить на классы
- Количество элементов в тестовом наборе можно варьировать

Спасибо за
внимание и ...

