Test2: жизнь после Test::More

Ликсанин Игорь
Backend разработчик,
АО «Объединенные Цифровые Сети»

Perl-Conf.Ru/25

О чём расскажу

1. Кто я
2. История тестирования в Perl и появление Test::More
3. Проблемы и ограничения Test::More
4. Что такое Test2 и зачем он нужен
5. Совместимость Test2 с существующим кодом
6. Практические примеры использования Test2
7. Выводы: стоит ли переходить на Test2

Ликсанин Игорь

Backend разработчик, АО «Объединенные Цифровые Сети»

- 8 лет в Perl: 4 года full-stack, 4 года только backend
- Перешёл в разработку из тестирования
- Контакты:
  Telegram: @iliksanin
  Email: i.liksanin@mail.ru

Краткая история тестирования в Perl

- Для perl был разработан TAP (*Test Anything Protocol*)
- Разработчики вручную писали строки вывода, например: `print "ok - Test passed\n";`
- Появились подпрограммы для автоматизации: `ok()` и подобные
- Стали появляться библиотеки объединиящие такие функции
- Возникали проблемы из-за их несовместимости

Краткая история тестирования в Perl

- Michael Schwern выпустил `Test::Builder` как совместимую основу, породив экосистему `Test::Builder`
- Schwern и другие пытались переработать `Test::Builder` для исправления недостатков, но проект закрыли
- Chad Granum выпустил `Test2` и обновил `Test::Builder`, превратив его в совместимый слой поверх `Test2`

Test::More - заслуженный ветеран

- Появился в 2001 году
- Стандарт де-факто для Perl-тестирования
  - Входит в ядро Perl (с версии 5.6.2)
  - Множество тестов написаны на нём
- Простой и понятный API
  - `ok(), is(), like(), done_testing()`
  - Легко освоить новичкам
- Покрывает 80% потребностей
  - Базовые проверки
  - Для сложных сценариев требуются дополнительные модули (`Test::Deep`, `Test::MockObject` и т.д.)

Проблемы Test::More

- Test::More построен вокруг единого глобального объекта Test::Builder. Весь код тесно связан:

```
# lib/Test/More.pm
sub is ($$;$) {
    my $tb = Test::More->builder;

    return $tb->is_eq(@_);
}
```

Проблема: невозможно изолировать части тестов или создать независимые тестовые контексты.

Проблемы Test::More

- Расширяющие функционал модули заменяли синглтон `Test::Builder`

```
1  subtest 'foo' => sub {
2      ok(1, 'bar');
3  };
4  # == v5.20 ==              == v5.42 ==
5  #     # Subtest: foo      # Subtest: foo
6  #     ok 1 - bar             ok 1 - bar
7  #     1..1                   1..1
8  # ok 1 - foo              ok 1 - foo
9  # 1..1                    1..1
```

Подробнее: `https://perldoc.perl.org/Test2::Transition`

Проблемы Test::More


```
tokuhirom@www4071uf:~/dev/Hok$ perl -Ilib eg/more2.t
        ok 1
        1..1
    ok 1 - should detect message with NG word
        ok 1
        1..1
    ok 2 - should not detect message without NG word
    1..2
ok 1 - MessageFilter
1..1
```

```
tokuhirom@www4071uf:~/dev/Hok$ perl -MTest::Pretty -Ilib eg/more2.t
  MessageFilter
      should detect message with NG word
        ✓    L22: ok($filter->detect('hello from foo'));
      should not detect message without NG word
        ✓    L25: ok(!$filter->detect('hello world!'));
```


Подробнее: https://github.com/tokuhirom/Test-Pretty/issues/25

Проблемы Test::More


```
1  package MyResponse;
2  sub new { bless { code => $_[1] }, $_[0] };
3  sub code { $_[0]->{code} }; 1;
4
5  package MyAssert;
6  use Test::More;
7  sub is_http_200 {
8      # local $Test::Builder::Level = $Test::Builder::Level + 1;
9      Test::More::ok( $_[0]->code == 200, $_[1] );
10 }; 1;
11
12 use open qw(:std :utf8);
13 use Test::More;
14
15 MyAssert::is_http_200( MyResponse->new(500), 'home' );
16 done_testing();
17 #   Failed test 'home'
18 #   at script.pl line 9.
```

Проблемы Test::More


```
1  package HelperA;
2  use Test::More;
3  sub init { Test::More->builder->no_diag(1) }  # меняет глобал
4  1;
5  package HelperB;
6  use Test::More;
7  sub init { diag 'я ожидал diag'; } # поведение уже изменено
8  1;
9
10 package main;
11 use Test::More;
12 HelperA::init();
13 HelperB::init();
14 ok 1;
15 done_testing();
16 # ok 1
17 # 1..1
```

Проблемы Test::More


```
# Миллионами в Test::More в рамках файла тесты лучше не гонять :)
for my $i (1 .. 1_000_000) {
    ok(1, "test $_");
    if ( $i == 1 || $i % 200_000 == 0 ) {
        show_mem();
    }
}
#            Test::More                      Test2::V0;
# tests   RSS     HWM     VMS      # tests   RSS     HWM     VMS
#        (MiB)   (MiB)   (MiB)              (MiB)   (MiB)   (MiB)
#     1   11.1    11.1    16.6     #     1   15.2    15.2    20.4
#  200k  121.9   121.9   127.3     #  200k   15.2    15.2    20.4
#  400k  232.8   232.8   238.1     #  400k   15.2    15.2    20.4
#  600k  343.2   343.2   348.6     #  600k   15.2    15.2    20.4
#  800k  454.6   454.6   459.9     #  800k   15.2    15.2    20.4
# 1000k  565.0   565.0   570.4     # 1000k   15.2    15.2    20.4
```

Неудобства Test::More

- При использовании utf8 с `Test::More` нужно не забывать сделать это с уважением

```
# Set the mode BEFORE anything loads Test::Builder
use open ':std', ':encoding(utf8)';
use Test::More;

# OR
# Modify the filehandles
my $builder = Test::More->builder;
binmode $builder->output,         ":encoding(utf8)";
binmode $builder->failure_output, ":encoding(utf8)";
binmode $builder->todo_output,    ":encoding(utf8)";

# OR (if Test::Builder used Test2)
use Test2::Plugin::UTF8;
```

Неудобства Test::More

- Отсутствие встроенной поддержки моков
- Отсутствие встроенной поддержки тестов исключений и предупреждений
- Отсутвие встроенной поддержки гибкого глубокого сравнения
- Рекомендации в документации по `Test::More` не использовать `Test::More` :)

Test2 - революция или эволюция?

- **Эволюция в использовании**
  - Совместимость: `Test::Builder` теперь слой над `Test2`
  - Привычные `ok/is/like/subtest` есть в `Test2::V0`
  - Улучшения ”из коробки”: красивый `Compare` с таблицами, `Subtest` с буферизацией
- **Революция в архитектуре**
  - События вместо глобального состояния
  - Плагины могут безопасно модифицировать поведение
  - Легко создавать библиотеки не опасаясь сломать чужой код

Что такое Test2

- Test2 - это фреймворк для написания тестовых инструментов
- Упрощает написание собственных расширений
- Предоставляет множество инструментов из коробки Test2::Suite

Подробнее: https://perldoc.perl.org/Test2::Suite

А что будет с Test::Builder?

- Сейчас Test::Builder - совместимый слой поверх Test2
- В таком виде с 10 мая 2016 года
- Можно смешивать инструменты из Test::Builder и Test2
- Вероятно он так и останется для совместимости со старым кодом

Test2::Suite

- Начиная с Perl 5.40 Test2::Suite входит в core
- Вскоре Test2::Suite также будет объединён с дистрибутивом Test-Simple
- Уже сейчас можно использовать в модулях ядра (со слов автора Test2)

Подробнее про экосистему Test2 от Chad Granum:
https://www.youtube.com/watch?v=DdSQkIfeoqU

Test::More и Test2::V0


```
1  use Test2::V0;
2
3  ok( 2 + 2 == 4, 'базовая математика' );
4
5  is( 42, 42, 'результат равен 42' );
6  isnt( 'ok', 'error', 'статус не ошибка' );
7
8  like( 'test@test.com', qr/\@/, 'содержит @' );
9  unlike( 'qw22', qr/123/, 'не содержит 123' );
10
11 done_testing();
```

```
ok 1 - базовая математика
ok 2 - результат равен 42
ok 3 - статус не ошибка
ok 4 - содержит @
ok 5 - не содержит 123
1..5
```

```
# Seeded srand with seed '20250926' from
     local date.
ok 1 - базовая математика
ok 2 - результат равен 42
ok 3 - статус не ошибка
ok 4 - содержит @
ok 5 - не содержит 123
1..5
```

Test::More и Test2::V0


```
1  use Test2::V0;
2
3  package MyClass;
4  sub new  { bless {}, shift }
5  sub save {1}
6  1;
7
8  package main;
9  my $obj = MyClass->new();
10 can_ok( $obj, qw(new save) );
11 isa_ok( $obj, 'MyClass' ); # отличается если больше x2- аргументов
12
13 done_testing();
```

```
ok 1 - MyClass->can(...)
ok 2 - An object of class 'MyClass' isa
    'MyClass'
1..2
```

```
ok 1 - MyClass=HASH->can(...)
ok 2 - MyClass=HASH->isa('MyClass')
1..2
```

Test::More и Test2::V0

Perl-Conf.Ru/25

```
1  use Test2::V0;
2
3  subtest 'An example subtest' => sub {
4      pass('pass');
5      ok(1, 'ok');
6  };
7
8  done_testing();
```

```
# Subtest: An example subtest
    ok 1 - pass
    ok 2 - ok
    1..2
ok 1 - An example subtest
1..1
```

```
ok 1 - An example subtest {
    ok 1 - pass
    ok 2 - ok
    1..2
}
1..1
```

Глубокие сравнения


```
1  use Test::More;
2
3  is_deeply([1,2,3], [1,2,3]);
4
5  # Failed test: order of elements
6  is_deeply(
7      { users => [qw/alice bob/]
       },
8      { users => [qw/bob alice/] }
9  );
10
11 # Failed test: extra field
12 is_deeply(
13     { age => 30, id => 123 },
14     { age => 30 }
15 );
16 done_testing();
```

```
ok 1
not ok 2
#   Failed test at 010.pl line 6.
#     Structures begin differing at:
#          $got->{users}[0] = 'alice'
#     $expected->{users}[0] = 'bob'
not ok 3
#   Failed test at 010.pl line 12.
#     Structures begin differing at:
#          $got->{id} = '123'
#     $expected->{id} = Does not exist
1..3
# Looks like you failed 2 tests of 3.
```

Глубокие сравнения


```
use Test::More;
use Test::Deep;

# Порядок не важен, кратности учитываются.
cmp_deeply( [ 1, 2, 2, 3 ], bag( 2, 3, 2, 1 ), 'same multiset' );

# Порядок не важен, элементы уникальны.
cmp_deeply( [ 1, 2, 2, 3 ], set(qw/1 2 3/), 'same set' );

# хешдолжен- содержать как минимум указанные пары.
cmp_deeply( {a=>1, b=>2}, superhashof({a=>1}), 'superhash ok' );

# сравнение по регулярному выражению.
cmp_deeply('Ivan Ivanov', re(qr/^Ivan /), 'string matches regex');

done_testing();
```

Глубокие сравнения


```
use Test2::Tools::Compare qw{
    is like isnt unlike
    match mismatch validator
    hash array bag object meta number float rounded within string subset bool
    in_set not_in_set check_set
    item field call call_list call_hash prop check all_items all_keys all_vals all_values
    etc end filter_items
    T F D DF E DNE FDNE U L
    event fail_events
    exact_ref
};

is(
    $some_hash,
    hash {
        field a => 1;
        field b => 2;
        field c => 3;
    },
    "Hash matches spec"
);
```


Подробнее: https://metacpan.org/pod/Test2::Tools::Compare

Глубокие сравнения


```
use Test2::V0;
is(
    [ 1, 2, 2, 3 ],
    bag {
        item 2;
        item 3;
        item 2;
        item 1;
    },
    'same multiset'
);
is(
    [ 1, 2, 2, 3 ],
    bag {
        item $_ for ( 2, 3, 2, 1 );
    },
    'same multiset'
);
```

Глубокие сравнения


```
use Test2::V0;
is(
    [ 1, 'Ivanov Ivan', {foo => 'bar'}, undef ],
    subset {
        item hash{
            field foo => E();
        },
    },
    'найден один элемент - хэш с ключом foo'
);
# T()     - true
# F()     - false (value must exist)
# D()     - defined
# U()     - undefined
# DF()    - defined but false
# E()     - exists
# L()     - defined and has length
# DNE()   - (D)oes (N)ot (E)xist
```

Глубокие сравнения


```
use Test2::V0;
is(
    {foo => 'foo', bar => 'bar'},
    hash {
        field foo => 'foo';
        field bar => match qr/a/;
        end;
    },
    "Hash matches expectations"
);

# end() - forbid any extra keys; must be last in the hash block
# etc() - allow extra keys; must be last in the hash block

done_testing();
```

Глубокие сравнения


```
1  use Test2::V0;
2
3  is(
4      { foo => 'bar', true => 0, false => 1, deep => { x => 5 } },
5      { foo => 'bat', true => T, false => F, deep => { x => 6 } },
6      "Nothing will match"
7  );
8  # Failed test 'Nothing will match'
9  # at snippets/015.pl line 3.
10 # +-----------+-----+---------+-------+-----+
11 # | PATH      | GOT | OP      | CHECK | LNs |
12 # +-----------+-----+---------+-------+-----+
13 # | {deep}{x} | 5   | eq      | 6     |     |
14 # | {false}   | 1   | FALSE() | FALSE | 3   |
15 # | {foo}     | bar | eq      | bat   |     |
16 # | {true}    | 0   | TRUE()  | TRUE  | 3   |
17 # +-----------+-----+---------+-------+-----+
18
```

Глубокие сравнения


```
is(
    bless( { foo => 'foo', bar => 'bar' }, 'My::Package' ),
    object {
        # Meta properties:
        prop blessed => 'My::Package';

        # Can check fields on underlying hash
        field foo => 'foo';
        field bar => match qr/a/;

        # Can check results of calling methods
        call foo => 'foo';

        # # Can check results of calling custom methods
        call sub { [ shift->list ] } => [ 'a', 'b', 'c' ];
    },
    "Object is as expected"
);
```

Исключения и предупреждения


```
1  use Test2::V0;
2
3  is(
4      dies { die 'xxx' },
5      match qr/xxx/,
6      "Got exception"
7  );
8
9  like(
10     dies { die 'xxx' },
11     qr/xxx/,
12     "Got exception"
13 );
14
15 ok(lives { 1 }, "did not die");
16
17 done_testing();
```

Исключения и предупреждения


```
use Test2::V0;

ok( warns { warn 'a' }, "the code warns" );
ok( !warns {1},         "The code does not warn" );
is( warns { warn 'a'; warn 'b' }, 2, "got 2 warnings" );

ok( no_warnings {1}, "code did not warn" );
like(
    warning { warn 'xxx' },
    qr/xxx/,
    "Got expected warning"
);
is(
    warnings { warn "a\n"; warn "b\n" },
    [ "a\n", "b\n" ],
    "Got 2 specific warnings"
);

```

Моки


```
1  use Test2::V0;
2  use v5.42;
3  package Calculator;
4  sub new ($class)          { bless {}, $class }
5  sub add ( $self, $x, $y ) { $x + $y }
6  1;
7
8  package main;
9  my $mock = mock 'Calculator' => (
10     add      => [ multiply => sub ( $self, $x, $y ) { $x * $y } ],
11     override => [ add      => sub {42} ],
12     set      => [ hello    => sub {"Hello"} ]
13 );
14 my $calc = Calculator->new();
15 is( $calc->add( 1, 1 ),      42,      'mock add' );
16 is( $calc->multiply( 2, 2 ), 4,       'mock multiply' );
17 is( $calc->hello(),          'Hello', 'mock multiply' );
18 done_testing();
```

Моки


```
1  use Test2::V0;
2  use v5.42;
3  package Calculator;
4  sub new (class)          { bless {}, class }
5  sub add ( self, x, y ) { x + y }
6  1;
7
8  package main;
9  my mock = Test2::Mock->new( class => 'Calculator' );
10 mock->before( 'add' => sub ( self, x, y ) { self->{_c}++ } );
11
12 my calc = Calculator->new();
13 calc->add( 1, 1 ) for 1 .. 42;
14
15 is( calc->{_c}, 42, 'before() counts add() calls' );
16 done_testing();
```

Подробнее: https://metacpan.org/pod/Test2::Mock

Прочее

- `Test2::Plugin::SRand`
  - Делает прогоны стабильными, но позволяет тестировать случайное поведение
  - Устанавливает seed равным текущей дате: `srand("YYYYMMDD")`
  - Seed можно переопределить переменной окружения `T2_RAND_SEED`
  - В подробном режиме или при падении сообщает использованный seed
  - Делает ”случайное” поведение воспроизводимым
  - Каждый день свой seed, изменения от ”рандома” всё ещё видны
  - Можно выключить поведение: `use Test2::V0 -no_srand => 1;`

Прочее

- `Test2::Plugin::UTF8`
  - ”UTF8 везде”
  - Импортирует прагму `utf8` в текущий файл
  - Устанавливает `STDOUT` в `utf8`
  - Устанавливает `STDERR` в `utf8`
  - Настраивает текущий форматтер на `UTF8` для всех дескрипторов
  - Применяет это во всех форматтерах
  - Можно выключить поведение:
    `use Test2::V0 -no_utf8 => 1;`

Резюме

- Для разработчиков приложений: `Test2::Suite`
  - `Test2::V0`: единый импорт и согласованный API
  - Сравнение структур
  - Моки
  - Хелперы: `warnings`, `lives`/`dies` и т.д.
  - Совместимость: `Test::More` поверх Test2 без боли
- Для разработчиков библиотек: `Test2::API`
  - Интерфейс для написания тестовых инструментов
  - Без синглтона: изоляция и предсказуемые хуки
  - Стабильные контракты вместо монкипатчинга

Посмотреть дополнительно

Better testing with Test2-Suite
https://www.youtube.com/watch?v=b2xIfBdcqb0

The Test2 Ecosystem
https://www.youtube.com/watch?v=DdSQkIfeoqU

Test2::Harness – Super charge your test runs
https://www.youtube.com/watch?v=uyqP6Prsq38

Документация: https://test-more.github.io/Test2-Suite

Спасибо за внимание!

Вопросы?

Perl-Conf.Ru/25