Ví dụ về 1 function đăng ký user mới

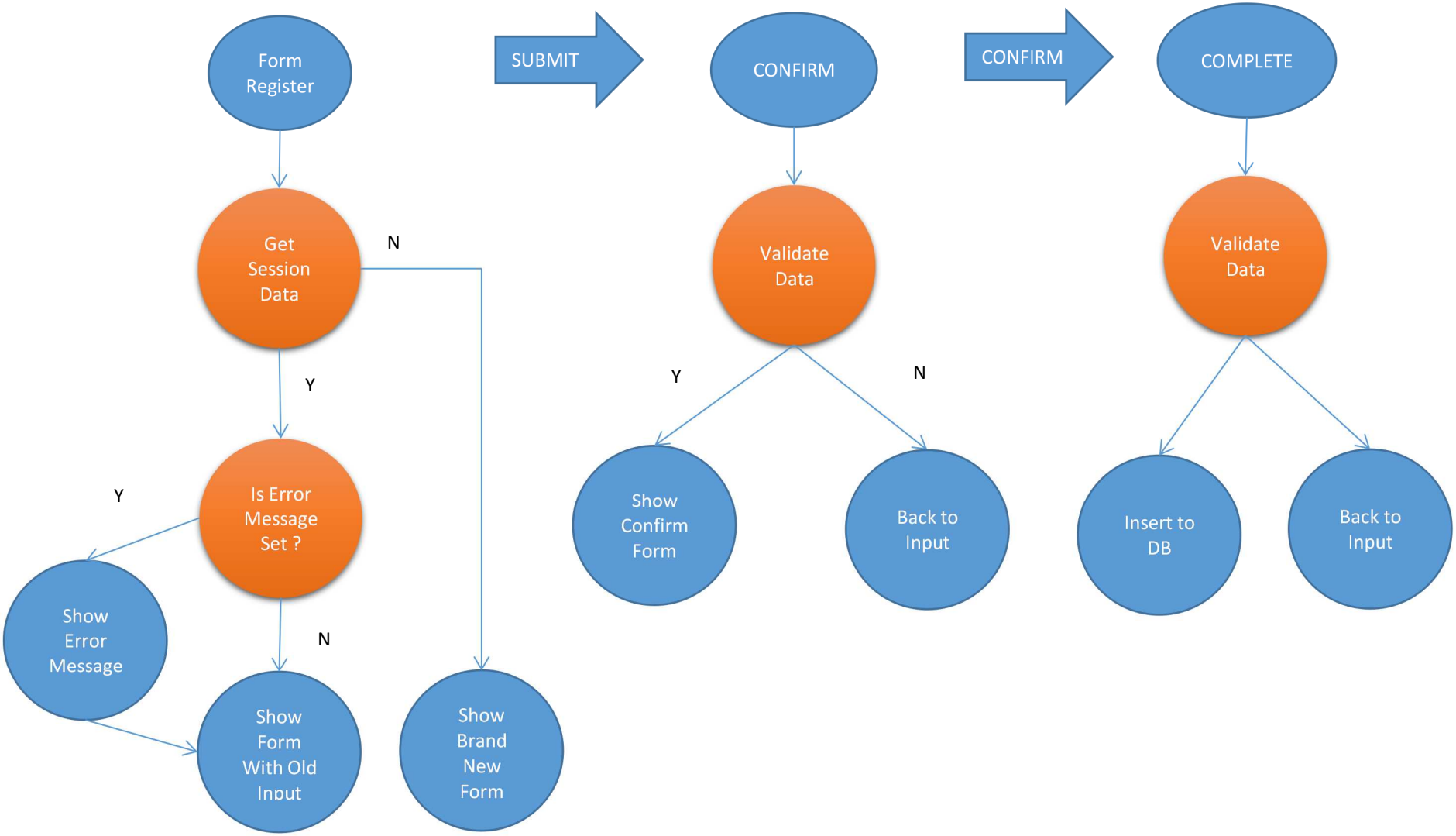
Vậy Unit Test cho trường hợp này sẽ như thế nào ?

Function gồm 3 site:

Form register : domain/regist-input(Controller & View)
Confirm : domain/regist-confirm (Controller & View)
Complete : domain/regist-complete (Controller & View & Model)

Bổ sung Unit Test cho function quan trọng là :

Validate Input ở Site Confirm và Complete
Insert to DB ở site Complete


Form Register
Get Session Data
N
Y
Is Error Message Set ?
Y
Show Error Message
N
Show Form With Old Input
Show Brand New Form
SUBMIT
CONFIRM
Validate Data
Y
N
Show Confirm Form
Back to Input
CONFIRM
COMPLETE
Validate Data
Insert to DB
Back to Input

Xác định các đường điều khiển (testcase) cần test:

Register Form :
- request -> has session data -> has error message -> show form
- request -> has no session data -> show form
- request -> has session data -> has no error message -> show form

Confirm Form :
- request -> valid data -> show confirm form
- request -> invalid data -> show input form

Complete Form :
- request -> valid data -> insert into DB
- request -> invalid data -> show input form

Xây dựng Test file và viết Code Test

1> Tạo ra Test file:

+ Run command :

php artisan make:test SomeThingTest

**** Lưu ý test file phải có đuôi là Test.php****

- SomeThingTest.php sẽ được tạo ra tại **app/tests/SomeThingTest.php**

2> Tạo function test, 1 function tương ứng với 1 test case:

```php
class TestRegistForm extends TestCase
{
    /**
     * A basic test example.
     *
     * @return void
     */
    public function testCaseOne()
    {
        $this->assertTrue(true);
    }
}
```

3> Thử Execute Test

+ Run command:

phpunit

Ta sẽ nhận được kết quả như sau:

```
$ phpunit
PHPUnit 4.8.36 by Sebastian Bergmann and contributors.

..

Time: 1.61 seconds, Memory: 10.00MB

OK (2 tests, 3 assertions)
```

4> Viết UT cho những case đã tìm được:

Register Form :

- Case 1 : request -> has no session data -> show form
- Case 2 : request -> has session data -> has no error message -> show form with old input & error message
- Case 3 : request -> has session data -> has error message -> show form

Confirm Form :

- Case 4 : request -> valid data -> show confirm form
- Case 5 : request -> invalid data -> show input form

Complete Form :

- Case 6 : request -> valid data -> insert into DB
- Case 7 : request -> invalid data -> show input form

Case 1: request -> has no session data -> show form
Mục tiêu: Test hiển thị form đăng ký

```
public function testRegistPage()
{
        $this->visit('/regist-input');
                $this->see("name");
                $this->see("email");
                $this->see("password");
                $this->see("password_confirmation");

}
```

Case 2: request -> has session data (old input) -> has no error message -> show form
Mục tiêu: test hiển thị dữ liệu đã input và không có báo error mesage

```
public function testRegistPageWithOldInput()
{
        //Giả lập dữ liệu session cho input
        $this->withSession(array(
                                '_old_input' => [
                                'name' => 'old name',
                                'email' => 'old email'
                                ],
        ));
        $this->visit('/regist-input');
        $this->seeInField("name", "old name");
        $this->seeInField("email", "old email");

}
```

Case 3: request -> has session data -> has error message -> show form
Mục tiêu: test hiển thị dữ liệu đã input và có kèm theo error message.

```
public function testRegistFormWithOldInputAndError()
{
        //Simulate error message
        $message = new Illuminate\Support\MessageBag();
        $message->add('name','error name');
        $message->add('email','error email');
        $message->add('password','error password');
        $errors_bag = new Illuminate\Support\ViewErrorBag();
        $errors_bag->put('default', $message);

        //Simulate old input & errors
        $this->withSession(array(
                        '_old_input' => [
                                        'name' => 'old name',
                                        'email' => 'old email'
                                        ],
                        'errors' => $errors_bag
                        ));
        $this->visit('/regist-input');
        $this->seeInField("name", "old name");
        $this->seeInField("email", "old email");
}
```

Case 4: request -> valid data -> show confirm form
Mục tiêu: test dữ liệu input [OK] chuyển sang confirm page.

```
public function testConfirmPagewithInputOK()
{
        //simulate token for post request
        $this->withSession(['_token' => 'dYSiA1gnXDpSb8rCXWRBNGkIJCLwgWqNQxEYcKOE']);
        //simulate post request
        $this->call('POST', '/regist-confirm', [
                '_token' => "dYSiA1gnXDpSb8rCXWRBNGkIJCLwgWqNQxEYcKOE",
                'name' => 'Taylor',
                'email' => 'baoluu@gmail.com',
                'password' => '123456',
                'password_confirmation' => '123456',
                ]);


        $this->seePageIs('/regist-confirm');
}
```

Case 5: request -> invalid data -> show input form
Mục tiêu: test dữ liệu input [NG] chuyển sang input page.

```
public function testRedirectWhenValidateFail()
{
        //simulate token for post request
        $this->withSession(['_token' => 'dYSiA1gnXDpSb8rCXWRBNGkIJCLwgWqNQxEYcKOE']);
        //simulate post request
        $this->call('POST', '/regist-confirm', [
                '_token' => 'test_token',
                'name' => '',
                'email' => '',
                'password' => '123456',
                'password_confirmation' => '123456',
                ]);

        $this->assertRedirectedTo('/regist-input');
}
```

Case 6: request -> valid data -> insert into DB
Mục tiêu: test dữ liệu input [OK] insert vào DB

```
public function testCompletePagewithInputOK()
{
        $name = 'bao';
        $email = 'baoluu@gmail.com';
        //simulate token for post request
        $this->withSession(['_token' => 'dYSiA1gnXDpSb8rCXWRBNGkIJCLwgWqNQxEYcKOE']);
        //simulate post request
        $this->call('POST', '/regist-complete', [
                '_token' => 'test_token',
                'name' => $name,
                'email' => $email,
                'password' => '123456',
                ]);

        $this->seeInDataBase('users', [
                                'name' => $name,
                                'email' => $email,
                                ]);
        $this->seePageIs('/regist-complete');
}
```

Case 7: request -> invalid data -> not insert into DB -> redirect input
Mục tiêu: test dữ liệu input [NG] và kiểm tra dữ liệu đó không được insert vào và chuyển sang input page.

```
public function testCompleteFailwithInputNG()
{
        $name = '';
        $email = 'baoluu@gmail.com';
        //simulate token for post request
        $this->withSession(['_token' => 'dYSiA1gnXDpSb8rCXWRBNGkIJCLwgWqNQxEYcKOE']);
        //simulate post request
        $this->call('POST', '/regist-complete', [
                '_token' => 'test_token',
                'name' => $name,
                'email' => $email,
                'password' => '123456',
                ]);

        $this->dontSeeInDatabase('users', [
                                'name' => $name,
                                'email' => $email,
                                ]);
        $this->followRedirects();

        $this->seePageIs('/regist-input');
}
```

Validation case:

Mục tiêu: test validation rule cho field: Name và hiển thị error message khi field không được nhập

Name : required - max:8
Email : required - unique: users, email
Password : required - confirmed - max 10

```
public function testNameEmpty()
{
        $data = [
            'name' => '',
            '_token' => 'dYSiA1gnXDpSb8rCXWRBNGkIJCLwgWqNQxEYcKOE'
        ];

        //test rule run right
        $rule = ['name' => 'required'];
        $validator = Validator::make($data,$rule);
        $is_fail = $validator->fails();
        $this->assertTrue($is_fail);

        //test controller run rule right
        $this->withSession(['_token' => 'dYSiA1gnXDpSb8rCXWRBNGkIJCLwgWqNQxEYcKOE']);
        $this->call('POST', '/regist-confirm', $data);
        $this->followRedirects();
        $this->seePageIs('regist-input');
        $error_message = $validator->messages()->all();
        foreach($error_message as $msg){
            $this->see($msg);
        }
}
```

Model Case:

Giả sử ta có model User với function như sau :

```
public function createUserWithPrefix($prefix, $name, $email, $password)
{
        User::create([
                'name' => $prefix.$name,
                'email' => $prefix.$email,
                'password' => bcrypt($password)
                ]);
}
```

Và chúng ta có thể gọi function **createUserWithPrefix** để test :

```
public function testCreateUserWithPrefix()
{
        $prefix = "your_prefix";
        $name = "name";
        $email = "email";
        $password = "123456";

        $user = new User();
        $user->createUserWithPrefix($prefix, $name, $email, $password);

        $this->seeInDatabase('users', ['name' => $prefix.$name, 'email' => $prefix.$email]);
        }
```

Redirect Case (Screen transition):

Giả sử ta có page input form với 3 field :

- Name
- Email
- Password

Nếu nhập dữ liệu hợp lệ sẽ đưa đến trang confirm và ngược lại sẽ show error message. Ta có thể test bằng cách tương tác với app như sau

```
public function testTypeFormToConfirmPage()
{
        $this->visit('regist-input')
        ->type('name','name')
```

```
        ->type('bao.luu@tctav.com','email')
        ->type('123456','password')
        ->press('Submit');
    $this->seePageIs('regist-confirm'); // kết quả mong đợi sẽ thấy trang confirm
}
```

Ta cũng có thể tương tác với file qua hàm :

```
$this->attach(string $absolutePath, string $element)
```

Có thể xem thêm các hàm interaction with application tại :

https://laravel.com/api/5.1/Illuminate/Foundation/Testing/InteractsWithPages.html

Một vài lưu ý khi ta viết UT:

Chuẩn bị database test riêng biệt:

Chúng ta nên có môi trường thực thi Unit Test độc lập :

- Tránh ảnh hưởng đến dữ liệu trong quá trình chạy production hoặc develop
- Dễ quản lý dữ liệu test
- Dữ liệu test là độc lập

Môi trường test có thể được config từ file **phpunit.xml**

- Các giá trị mặc định được config sẵn đã phù hợp để test
- Chúng ta chỉ nên config thêm phần database test độc lập :

Config Phpunit.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<phpunit backupGlobals="false"
backupStaticAttributes="false"
bootstrap="bootstrap/autoload.php"
colors="true"
convertErrorsToExceptions="true"
convertNoticesToExceptions="true"
convertWarningsToExceptions="true"
processIsolation="false"
stopOnFailure="false">
<testsuites>
    <testsuite name="Application Test Suite">
        <directory suffix="Test.php">./tests</directory>
        <file>./test/UserModelTest.php</file>
    </testsuite>
</testsuites>
<filter>
    <whitelist processUncoveredFilesFromWhitelist="true">
    <directory suffix=".php">./app</directory>
    <exclude>
        <file>./app/Http/routes.php</file>
    </exclude>
</whitelist>
</filter>
<php>
<env name="APP_ENV" value="testing"/>
<env name="DB_DATABASE" value="laravel51_test"/>
<env name="CACHE_DRIVER" value="array"/>
<env name="SESSION_DRIVER" value="array"/>
<env name="QUEUE_DRIVER" value="sync"/>
</php>
</phpunit>
```

Giả lập các tài nguyên cần thiết

Giả lập dữ liệu database :

- Để giả lập dữ liệu cho 1 test case, ta có thể sử dụng modelFactory của laravel (https://laravel.com/docs/5.1/testing#model-factories/)
- Có thể sử dụng migration để migrate
- Cũng có thể sử dụng chức năng seeding của laravel. Cách gọi lệnh artisan :

```
$this->artisan('migrate');
$this->artisan('db:seed',['--class' => 'SeederClass']);
```

- Nếu cần setup 1 tập dữ liệu cho 1 class test, ta có thể sử dụng hàm setUp

```
public function setUp()
{
    parent::setUp();
    $this->artisan('db:seed',['--class' => 'TestingSeeding']);
}
```

- Laravel cung cấp 2 trait để xử lý việc rollback db hoặc đưa các testcase vào transaction.

```
class CompleteSiteTest extends TestCase
{
    use DatabaseMigrations;
    use DatabaseTransactions;
```

Giả lập dữ liệu session :

- Để giả lập dữ liệu session cho 1 test case, ta sử dụng mock event của laravel

```
$this->withSession(['foo' => 'bar'])
```

Giả lập dữ liệu từ request :

- Để giả lập request với dữ liệu

```
$response = $this->call('GET', '/');
```

or

```
$response = $this->call('POST', '/user', ['name' => 'Taylor']);
```

Giả lập quyền hạn user :

- *Cách 1* : disable check quyền qua middleware bằng cách sử dụng :

```
$this->withoutMiddleware();
```

- *Cách 2* : giả lập đăng nhập với user có quyền hạn muốn check

```
$this->actingAs($user);
```

Tài liệu nghiên cứu:

Laravel v5.1 đã tích hợp và config sẵn PHPUnit cho chúng ta.

Document có thể xem tại :
https://laravel.com/docs/5.1/testing

Testing Api có thể xem tại :
https://laravel.com/api/5.1/Illuminate/Foundation/Testing.html
https://laravel.com/api/5.1/Illuminate/Foundation/Testing/CrawlerTrait.html

========== The End ==========