EC-CUBE3 受入テスト Ver.1.0

作成: オフィス ORA 六藤
[http://www.office-ora.com/ kmutoh@office-ora.com]
2016/1/31

はじめに

> **受け入れテスト (Acceptance Testing)**
> システムが、ユーザーのニーズ、要件、ビジネス・プロセスを満足するかをチェックするための公式のテスト。このテストにより、システムが承認基準を満たしているかを判断したり、ユーザー、顧客、その他の認可主体がシステムを承認するかしないかを判定できる

EC-CUBE3 を使って、EC システムを構築しそれを公開・納品する際に「承認」を獲得する為には受け入れテストがとても重要です。

EC-CUBE3 の公式な受け入れテストおよびその実行環境の容易な構築方法をつくりましたので、そのまとめをここに記述します。

今回つくった受入テストを総合して以下「EC-CUBE3 受入テスト Ver.1.0」と記述します。

EC-CUBE3 の受け入れテスト概要

テスト項目

EC-CUBE/eccube3-doc (https://github.com/EC-CUBE/eccube3-doc) の IntegrationTest ディレクトリに EC-CUBE3 のデフォルト状態に関するテスト項目がまとまっています。

「EC-CUBE3 受入テスト Ver.1.0」では、ここに記述されているテスト項目をカバーしました。

EC-CUBE3 を使って EC システムを構築する際には、そのカスタマイズおよび拡張内容に沿って、このテスト項目を更新し、「EC-CUBE3 受入テスト Ver.1.0」のテストコードを更新・実行しましょう。

テストフレームワーク

「EC-CUBE3 受入テスト Ver.1.0」では Codeception (http://codeception.com/) という PHP でかかれたテストフレームワークを利用しています (現時点でバージョンは 2.1.4 です) 。

テストにつかうクライアントブラウザ

「EC-CUBE3 受入テスト Ver.1.0」では、PhantomJS (http://phantomjs.org/) を利用します。

Codeception テストフレームワークは、Selenium/PhantomJS/PHP Browser をサポートしています。PHP Browser は機能的に不十分ですし、Selenium はテストに時間がかかる為、PhantomJS を採用しました。

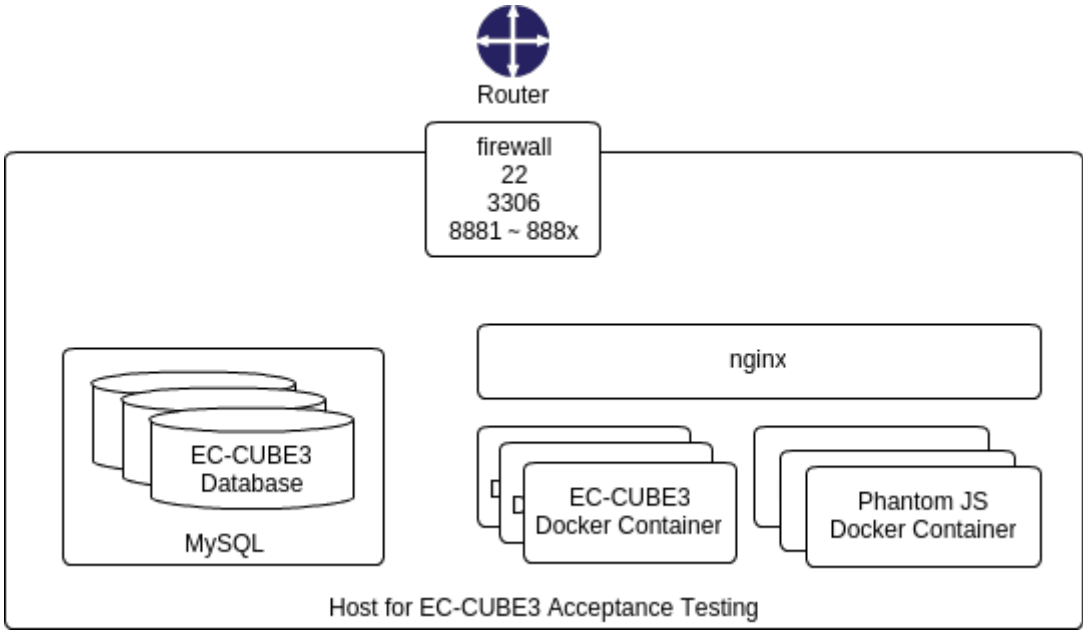
EC-CUBE3 の対象バージョン

「EC-CUBE3 受入テスト Ver.1.0」では、EC-CUBE3 Ver.3.0.7 を対象にテストコードが書かれています (今後、最新の EC-CUBE3 に対応してアップデートされていくでしょう) 。

ご利用の EC-CUBE3 のバージョンに合わせて、テストコードを更新して利用してください。

導入からテスト実行まで

「EC-CUBE3 受入テスト Ver.1.0」の実行環境構成



以下の様な構成を前提にしています。

前提条件

受入テストを実行するホストには以下がインストールされている必要があります。

- PHP 5.4.14 以上 (テストする対象の EC-CUBE3 が動作する PHP 環境である必要があります)
- git
- Docker (動作確認：ver.1.9.1)
- nginx (動作確認：ver.1.8.0)

また、EC-CUBE3 が利用するデータベースは、このホストに存在する必要はありませんが、今回は便宜的に同一ホストに MySQL をインストールしていることとします。

受入テスト実行ホストは Firewall の裏側にあるケースが多いと思いますが、その場合は以下のポートが開放されていることとします。

- 8881 ~ 888x (外部から EC-CUBE3 Docker Container にアクセスするポート群。"x"は用意する EC-CUBE3 Docker Container の数に対応します)

- 3306 (MySQL がホスト内にあるので。PostgreSQL など他のデータベースエンジンを利用する場合はそれが使っているポートを開放します。外にある場合は必要なしです)
- 22 (受入テストには直接関係ありませんが、ssh アクセス用に開けときます)

環境作成とテスト実行

環境作成は以下の手順で行います。

1. 「EC-CUBE3 受入テスト Ver.1.0」のインストール
2. env/build_env.sh による環境整備
3. データベースの用意
4. install.php によるデータベースの初期化
5. 「EC-CUBE3 受入テスト Ver.1.0」の設定および設定の確認
6. 受入テスト実行

以下にそれぞれのステップについて記述します。

1. 「EC-CUBE3 受入テスト Ver.1.0」のインストール

- Git で EC-CUBE/eccube-codeception を clone します
- clone したディレクトリに移動して以下を実行します (vendor ディレクトリができます)

```
$ curl -sS https://getcomposer.org/installer | php
$ php composer.phar install
```

2. env/build_env.sh による環境整備

「EC-CUBE3 受入テスト Ver.1.0」の実行環境をつくるには eccube-codeception/env 以下にある build_env.sh スクリプトを使います。

- eccube-codeception/env ディレクトリに移動
- ./build_env.sh を実行 (以下の質問に答えていく)
  - **“PHP のバージョン指定[5/5|5.6|7.0]”**
    ECCUBE3 Docker Container にインストールする PHP のバージョンを指定 (例：5.6)
  - **“codeception テストを動かすユーザー”**
    ホストで受入テスト実行するユーザーアカウント名 (ホストによって違うので確認して入力)
  - **“コンテナの数 (並列テストする数) を指定[1 以上]”**
    テストを分割して同時並列に実行する場合に用意する EC-CUBE3 および PhantomJS のコンテナ数
  - **“このホストのホスト名(IP アドレス)”**
    「EC-CUBE3 受入テスト Ver.1.0」を動作させるホストのホスト名か IP アドレス。現状の

「EC-CUBE3 受入テスト Ver.1.0」ではホストは外部からアクセス可能である必要があります。

- **“ECCUBE3 のタグ[3.0.0 <]”**
  テスト対象の EC-CUBE3 のバージョンを指定します。現状の「EC-CUBE3 受入テスト Ver.1.0」は 3.0.7 にのみ対応していますので、3.0.7 を入力してください（「EC-CUBE3 受入テスト Ver.1.0」をアレンジした場合はご利用の EC-CUBE3 のバージョンを指定します）
- **“ECCUBE3 をインストールするディレクトリ”**
  ECCUBE3 Docker Container は、EC-CUBE3 をインストールするディレクトリをホストと共有していなければなりません。なので、ホスト上のどのディレクトリを ECCUBE3 Docker Container と共有するか指定してください（例：/home/testuser/）。
  指定するディレクトリの最後には必ず”/”をつけてください。

build_env.sh は以下のことを行います。

- PHP がインストールされているかどうかチェック
- Docker がインストールされ、起動されているかどうかチェック
- git がインストールされているかどうかチェック
- PhantomJS の Docker image 作成（あれば既存のものを使う “davert/pahntomjs-env:latest”）
- ECCUBE3 Docker Container の Docker image 作成（あれば既存のものを使う “ec-cube/ec-cube3:latest” 作成には eccube-codeception/env/docker/Dockerfile を使うので、多少時間がかかる）
- PhantomJS のコンテナ作成および起動（コンテナはあれば既存のものを使う “client1 ~ x”という名前でコンテナが作られる）
- EC-CUBE3 の指定されたホスト上ディレクトリへの clone/composer install および必要なディレクトリ権限変更（”[指定ディレクトリ]cube3-1 ~ x”という名前で clone される）
- ECCUBE3 Docker Container のコンテナ作成および起動（コンテナはあれば既存のものを使う “cube3-1 ~ x”という名前でコンテナが作られる）

実行が終わると、「指定された数だけ ECCUBE3 Docker Container と PhantomJS コンテナが作成され立ち上がっている」状態になり、eccube-codeception/env 以下に nginx 用の configure ファイルが作成されますので、これらを nginx の起動時に読み込む様に配置し nginx を再起動してください。

各 ECCUBE3 Docker Container には、外部から

```
http://[ホスト名 or IP アドレス]:888(1 ~ x)

例： http://www.eccube-test..com:8881
```

で、ブラウザからアクセスできるようになります（下記「3.データベースの用意」以下を実行してからです...この時点でアクセスしてもエラーになるだけです）。

3. データベースの用意

利用する ECCUBE3 Docker Container の数分、MySQL にデータベースを作成しておきます。

```
mysql> CREATE DATABASE eccube01 DEFAULT CHARACTER SET 'utf8';
```

また、MySQL の場合、作成したデータベースへのアクセスに使うユーザーとそのユーザーの権限（外部からアクセス可能）を設定する必要があります。

```
mysql> GRANT ALL PRIVILEGES on eccube01.* TO testuser@'%' IDENTIFIED BY
'password' with grant option;
mysql> FLUSH PRIVILEGES;
```

4. install.php によるデータベースの初期化

データベース作成が完了したら、EC-CUBE3 の初期化を実施します。

「2. env/build_env.sh による環境整備」で作成した ECCUBE3 Docker Container に、外部からブラウザでアクセスします。

```
http://www.eccube-test..com:8881/install.php
```

画面の指示にしたがって初期化してください。

「管理画面のログイン ID・パスワード」「管理画面のディレクトリ名」は、後に「EC-CUBE3 受入テスト Ver.1.0」の設定で使うのでメモしておくといいです。

「データベースの設定」は、「3. データベースの用意」で作成したデータベースと、ユーザー・パスワードを指定します。

5. 「EC-CUBE3 受入テスト Ver.1.0」の設定および設定の確認

「EC-CUBE3 受入テスト Ver.1.0」の設定を行います。

確認すべきファイルは 2 つあります。それぞれについて以下に記述します。

- tests/acceptance.suite.yml
    - modules:config:Db セクション（env オプションを使わない場合に使われる設定）
    - modules:config:WebDriver セクション（env オプションを使わない場合に使われる設定）
    - env セクション
        - env セクションは複数のコンテナを用意し並列テスト実行する場合に必要になります
        - env:admin や env:front のように環境に名前をつけ、各テストが指定された環境に沿ったデータベース・PhantomJS コンテナ・ECCUBE3 Docker Container を使うようにそれぞれに以下を設定します

          デフォルトでは、3 つのコンテナを使い「admin01/admin02/front」の環境設定が利用できるように記述されていますので、これを参考にして設定してください。

            - modules:config:Db:dns

        - modules:config:WebDriver:port
        - modules:config:WebDriver:url
- tests/acceptance/config.ini
    - eccube_path (build_env.sh で作成された EC-CUBE3 のホストでのディレクトリ 例：/home/testuser/cube3-1/)
    - admin_user (install.php で指定した管理画面ログイン ID)
    - admin_password (install.php で指定した管理画面パスワード)
    - hostname (ECCUBE Docker Container のホスト名とポート 例：http://www.eccube-test.com:8881)
    - db / dbhost / dbport / user / password / charset (データベース接続情報)

> **メモ**
>
> 受入テストを実行する場合は「4. install.php によるデータベースの初期化」を毎回行う必要があります。
>
> しかし、テストを記述する時など「毎回 install.php するのは時間がかかって手間だ...」という場合があります。
> そういう時は、tests/acceptance.suite.yml の modules:config:Db に dump を設定します。
>
> modules:config:Db:dump には、mysqldump で作成したダンプファイルへのパスを指定します。
> ダンプは、tar などで圧縮せず、生の sql ファイルである必要があります。
> ダンプファイルの置き場所は、codeception では eccube-codeception/test/_data/が推奨されています。
> (例：tests/_data/cube1.dump.sql _data　ディレクトリならこの様に相対パスで記述できます)

6. 受入テスト実行

さあ、いよいよテストの実行です。

ホストのコマンドプロンプトで以下のコマンドを使います。

```
$ cd [eccube-codeception のディレクトリ]
$ php vendor/bin/codecept run –env admin01
```

このコマンドでは、「環境 admin01 の設定を使って、すべてのテストを一括で実行」します。

[eccube-codeception のディレクトリ]tests/acceptance/ディレクトリには、実行されるテストが php クラスファイルとして存在します。

フロントと管理画面を分けて実行するには以下の様にします (コンテナを 2 つ用意していれば、コマンドプロンプトを 2 つ用意して同時に実行可能です)。

```
$ php vendor/bin/codecept run -g admin –env admin01
$ php vendor/bin/codecept run -g front –env front
```

すべてのテストが正常に終わる場合、フロントは約 10 分くらい、管理画面は約 70 分くらいかかります。

ここまでくると管理画面のテストも分割したくなりますね。そういう場合は以下の様にします (コンテナを 3 つ用意していれば、コマンドプロンプトを 3 つ用意して同時に実行可能です) 。

```
$ php vendor/bin/codecept run -g admin01 –env admin01
$ php vendor/bin/codecept run -g admin02 –env admin02
$ php vendor/bin/codecept run -g front –env front
```

admin01 グループは約 40 分、admin02 グループは約 30 分くらいかかります。

> **メモ**
>
> 「EC-CUBE3 受入テスト Ver.1.0」の動作確認は、AWS EC2 t2.micro インスタンス (メモリ 1G) と Nifty Cloud small2 インスタンス (メモリ 2G) で行われました。
> もちろん、メモリ 2G の Nifty Cloud の方が早くテスト実行できるわけですが、上述のテストにかかった時間の数値は Nifty Cloud での実測に基づくものです。

実行に関する FAQ

なぜ ECCUBE3 Docker Container なの？

これは EC-CUBE3 の設計によります。

EC-CUBE3 では「DI コンテナ」として Pimple を利用しています。
EC-CUBE3 の設定や、各ライブラリやオブジェクトは、この DI コンテナに格納され共有されつつ EC-CUBE3 の実行時に使われます。

EC-CUBE3 の前提条件である Apache で mod_php を使って複数の EC-CUBE3 を動かすことを考える場合、各 EC-CUBE3 で DI コンテナが共有されてしまうのでうまく動作しません。

ここで、Apache で mod_fastcgi を使い EC-CUBE3 のプロセスを分散させるか、今回採用したように Docker で Apache ごと分散させるか、、2 つに 1 つの選択肢になるわけですが、今回は Docker を使ってみました (少々大袈裟な感じはしますが...) 。

なので、ECCUBE3 Docker Container を使いたくなければ mod_fastcgi でプロセス分散する形を試してみてください。

PhantomJS については、このまま Docker コンテナを利用することをおすすめします。

並列テスト実行しないので、ECCUBE3 Docker Container いらない？

並列テスト実行しなくても、ECCUBE3 Docker Container と PhantomJS のコンテナを 1 つずつ用意して受入テストする。。という形を「EC-CUBE3 受入テスト Ver.1.0」では考えています。

ただし、これは以下の形に置き換えられます。

- nginx を proxy とし ECCUBE3 Docker Container に外部からアクセスできるようにする代わりに、、、ホストに Apache をインストールし、テスト対象の EC-CUBE3 を 1 つ可動させる

後は、PhantomJS の Dokcer コンテナと、データベースが 1 つあれば「EC-CUBE3 受入テスト

Ver.1.0」のテストは実行できます（acceptance.suite.yml / config.ini の内容を適切に設定してください）。

単独テストのみなら、この選択肢は悪くない選択だと思います。

受入テスト実行するホストを外部に晒したくないです...

「導入からテスト実施まで - 前提条件」で ECCUBE3 Docker Container へのアクセス用ポートと、MySQL へのポートを開放しておくと記述しました。

もちろん、できればこれは外部に晒さないでおきたいですよね。

基本的には、以下のような形がとれれば、今回示した形でなくても「EC-CUBE3 受入テスト Ver.1.0」を実行することは可能です。
用意した VPC などの内側ですべてがまかなえるように試してみてください。

- ECCUBE3 Docker Container から、MySQL にアクセス可能
- PhantomJS Docker コンテナから nginx を通じて各 ECCUBE3 Docker Container にアクセス可能

受入テスト記述

[eccube-codeception のディレクトリ]tests/acceptance/ディレクトリには、実行されるテストが php クラスファイルとして存在します。

これらをベースにして、あなたの EC-CUBE3 をベースにした EC システムの仕様に沿ったテストを作成し、リリース前に受入テスト実行してください。

piccagliani さんが codeception のドキュメントを日本語化してくれています（http://piccagliani.github.io/Codeception.docs.ja_JP/ git リポジトリ：piccagliani/Codeception.docs.ja_JP https://github.com/piccagliani/Codeception.docs.ja_JP）。

codeception について知るには、まず以下を熟読しましょう。

- 1. イントロダクション http://piccagliani.github.io/Codeception.docs.ja_JP/01-Introduction.html
- 2. はじめに http://piccagliani.github.io/Codeception.docs.ja_JP/02-GettingStarted.html
- 3. 受け入れテスト http://piccagliani.github.io/Codeception.docs.ja_JP/03-AcceptanceTests.html
- 6. テストコードの再利用 http://piccagliani.github.io/Codeception.docs.ja_JP/06-ReusingTestCode.html
- 7. 高度な使用法 http://piccagliani.github.io/Codeception.docs.ja_JP/07-AdvancedUsage.html

その上で[eccube-codeception のディレクトリ]tests/acceptance/のテストをながめてみてください。

また、テストを記述する上でよく使う codeception の WebDriver モジュール・Assets モジュールのリファレンスは以下にあります。

- WebDriver http://codeception.com/docs/modules/WebDriver
- Asserts http://codeception.com/docs/modules/Asserts

これらを駆使してテストを記述していってください。

_bootstrap.php

[eccube-codeception のディレクトリ]tests/acceptance/_bootstrap.php には、テスト実行の前処理が記述されています。

以下の処理がされます。

1. config.ini の読み込み
    - 読み込み
    - env オプションによって、config.ini の指定 env 用設定を読み込み＆上書き
2. Fixture の作成
    - EC-CUBE3 を実行し、EC-CUBE3 の初期化を行う
    - Doctrine を通じてテスト用 Fixture を作成（会員・注文・商品を config.ini の設定数分つくる）
3. config.ini と EC-CUBE3 の設定をテスト内で利用できるように Fixture として格納
4. EC-CUBE3 の BaseInfo/Category/News を Doctrine を通じて読み込み、テスト内で利用できるように Fixture として格納

あなたの EC-CUBE3 をベースにした EC システムの仕様に沿ったテストを記述する際、これらとは別に Fixture を用意したり、設定を追加したりする場合は、この_bootstrap.php にその準備を記述しましょう。
テスト記述において Fixture の用意はとても重要です。

AcceptanceTester.php

[eccube-codeception のディレクトリ]tests/_support/AcceptanceTester.php も、テスト記述にとって重要な役割を果たします。

これは、あなたが作成する[eccube-codeception のディレクトリ]tests/acceptance 以下の PHP のテストクラスの各テストメソッドで引数として渡される”$I”です。

各テストで共通のテストコードなどをまとめておきたい場合は、この AcceptanceTester.php にメソッドとして記述しておくと便利です。

デフォルトでは、以下のメソッドを用意しました。

- loginAsAdmin() : 管理者として管理画面にログインします
- logoutAsAdmin() : 管理画面からログアウトします
- goToAdminPage() : 管理画面に移動します

- loginAsMember() : 会員としてショップサイトにログインします
- logoutAsMember() : ショップサイトからログアウトします
- setStock() : 指定された商品の在庫数を設定します
- buyThis() : ショップサイトの商品詳細ページ上でこのメソッドを呼ぶと指定された個数商品を購入します
- makeEmptyCart() : ショップサイトのカートを空にします

テスト記述に関するメモ

- **javascript の alert()によるポップアップは PhantomJS ではハンドリングできない**
  EC-CUBE3 の各所では alert()が使われています。これは PhantomJS ではハンドリングできないのでテスト不可能です。Selenium はこれをハンドリングできるのでテスト可能です。
  基本的には、Selenium に対する PhantomJS でのテストスピードの優位性は捨てきれませんので、どうしてもこれをテストしたい場合は、alert()の部分を javascript や HTML5 でテスト可能な様に置き換えることをおすすめします
- **フォームのテストは遅い**
  これは WebDriver の性能の問題と思います。。WebDriver で用意されているメソッド fillField()や selectOption()するとかなり遅いです。
  遅いといって、これを省略するわけにはいきません。
  本質的な解決方法ではありませんが、フォームの入力・選択要素に明確な id 属性を指定することで多少の効果があるようには思います。
- **クラスは個別にテストできるようにする**
  各テストクラスは、デフォルトでは相互の依存性はありません。
  （ただし、クラス内のテストメソッドについては、数ヶ所、同じクラス内の他のテストコードに依存性をもつものが存在します）
  クラス間の相互依存性は、テストの効率を考えると不利ですので、あなたの EC-CUBE3 をベースにした EC システム用にテストを拡張・追加する場合もこれを維持するように記述することをおすすめします。
  特に分散テスト実行を考えている場合は重要なポイントです。