報道各位

2020年5月21日
株式会社インプレスR&D
https://nextpublishing.jp/

バックエンドにExpressを使ってAzureに公開！

『Vue CLIがわかる！使える！TDDでつくるアプリ開発入門』発行

技術の泉シリーズ、5月の新刊

インプレスグループで電子出版事業を手がける株式会社インプレスR&Dは、『Vue CLIがわかる！使える！TDDでつくるアプリ開発入門』(著者:窓川ほしき)を発行いたします。

最新の知見を発信する『技術の泉シリーズ』は、「技術書典」や「技術書同人誌博覧会」をはじめとした各種即売会や、勉強会・LT 会などで頒布された技術同人誌を底本とした商業書籍を刊行し、技術同人誌の普及と発展に貢献することを目指します。

『Vue CLIがわかる！使える！TDDでつくるアプリ開発入門』
https://nextpublishing.jp/isbn/9784844378532

著者:窓川ほしき
小売希望価格:電子書籍版 1600円(税別)／印刷書籍版 2000円(税別)
電子書籍版フォーマット:EPUB3／Kindle Format8
印刷書籍版仕様:B5判／カラー／本文100ページ
ISBN:978-4-8443-7853-2
発行:インプレスR&D

＜＜発行主旨・内容紹介＞＞

本書は、テスト駆動開発(TDD)でVue.jsによるWebアプリケーションをAzureに公開するまで、の手順を解説しています。

Vue.jsをCDN版で少し触れてみた人を対象に、①Vue-CLIへの移行、②Vue-CLIでのテスト駆動開発、③Expressでのテスト駆動開発を説明します。

「webpackとBabelなどについて詳しく知らなくても、とりあえず使えるようになる」ことを目指します。

(本書は、次世代出版メソッド「NextPublishing」を使用し、出版されています。)

CDN 版で作成したソースコードを Vue CLI 版へ移行する方法を解説

しその一方で、ブラウザー環境はファイル単位でのスコープ機能が無く、「被テスト対象を限定してテストをする」ことが困難です。

Vue.jsのCLI版を利用することで、「Node.js環境でテストして、ブラウザー実行時にはブラウザー環境向けにトランスパイルして出力する」ことが可能となります。Node.js環境であれば、ファイル単位のスコープ機能があり、テスト駆動開発を支援する機能も利用し易いです。

CDN版で作成したソースコードをVue CLI版へ移行する際のポイントは次のとおりです。

- htmlファイルで「`id="id_app1"`」としていたdivブロックを、vueファイルのtemplateタグ直下へ移動
- javascript部分は、vueファイルのscriptタグ配下へ移動
  - Vueインスタンス化するdivブロックとして指定したidは削除（「`id="id_app1"`」などを削除）
  - elタグに代わってnameタグを用いる
  - 「`data : {}`」の部分を、「`data : function(){return: {}; }`」のように、関数の戻り値としてオブジェクトを返却するように変更
- css部分は、styleタグ配下へ移動

CLI版への具体的な移行手順は以下の通りです。

1．Vue CLI環境をNode.js上に導入する
2．CDN版で作成した3ファイルを、MyClient.vueファイルにまとめる（ファイル名は任意）
3．Vue CLIのスタートファイルとなるApp.vueに、「MyClient.vueを表示して」と記述
4．devトランスパイルして（開発時の動作確認用）、ローカルのブラウザーで表示を確認
5．buildトランスパイルして（公開用）、実際にHTTPサーバーに配置して、ブラウザー表示を確認

次の節から、実際に先ほどのhtmlのコード（リスト1.1）とjavascriptのコード（リスト1.2）、Cascading Style Sheetのコード（リスト1.3）を、Vue CLIのvueファイルへ移行していきます。

1.2　Vue CLI環境を導入

作成物を格納したい任意のフォルダに移動し、Node.js環境でnpmのpackage.jsonを作成します。これは、通常通りに`npm init`で適当に作成してください。例えば筆者は次のように、名称と作者名、ライセンスのみ入力しました。

```
C:\[My]GitHub\tbf07-sample>npm init

package name: (tbf07-sample) tbf07-sample
version: (1.0.0)
description:
entry point: (index.js)
test command:
git repository:
keywords:
author: hoshimado
license: (ISC) MIT
Is this OK? (yes) yes

C:\[My]GitHub\tbf07>
```

以下のコマンドを実行してVue CLIをインストールします（Vue CLIのVersion.3系です）。異なる端末上でも簡単に再導入できるようにすることを目的に、本書では「ローカル」にインストールします。（※「グローバル」にインストールしたい場合は「-g」オプションを付けてください）

```
npm install @vue/cli --save-dev
```

なぜ、ローカルにインストール？

他の環境で、Githubなどから取得してきたとします。その環境ではVue CLIが未導入だとします。ローカルにインストールしてpackage.jsonに記録しておくことで、1行のコマンド「`npm install`」を実行するだけで、開発環境を再現することが出来るからです。

なぜ、「-save-dev」でインストール？

`@vue/cli`コマンドを用いるのは開発環境のみだから、です。公開用にトランスパイルした後は、Vue CLIが必要な場面はありません。必要の無いものを、Deploy環境にインストールしたくないからです。

つづいて、以下のコマンドを実行して、Vue CLIのスケルトンを作成します。ここで作成するものは「プロジェクト」とも呼ばれ、Vueフレームワークの構成ファイル一式です。プロジェクト名は、ここでは「cli-vue」とします。（※先述の`@vue/cli`を「グローバル」にインストールした場合は「vue create cli-vue」で実行します）

```
    node_modules\.bin\vue create cli-vue
```

コマンドを実行すると、最初に次の質問（リスト1.4）を聞かれます。

前章での解説を元にテスト駆動開発（TDD）で機能を追加

第2章　Vue CLIでテスト駆動開発する（TDD）

第1章「Vue.jsのCDN版からVue CLI版へ移行」にて、CDN版のToDoのWebアプリをVue CLIへ移行することが出来ました。本章では、これをベースにテスト駆動開発で機能を追加していきます。

なお、本章でもフォルダ「cli-vue」配下を起点としてファイルパスを記載します。「`src\components\MyClient.vue`」と書いた場合は「`\cli-vue\src\components\MyClient.vue`」を意味します。

Vue CLI環境でユニットテストを実行するには、Vue CLIのプロジェクトを作成する際に、「Unit Testing＞Mocha + Chai」を指定する方法が簡単です。具体的な指定方法は、第1章「Vue.jsのCDN版からVue CLI版へ移行」のリスト1.5を参照ください。

2.1　MochaとChaiによるテスト（検証）の仕組み

第1章「Vue.jsのCDN版からVue CLI版へ移行」のようにしてプロジェクトを作成すると、テストのサンプルが予め作成されています。ToDOアプリとして作成追加した`src\MyClient.vue`をいったん忘れて、デフォルトで生成されていた`src\HelloWorld.vue`ファイルに対するVue CLIのデフォルトのテストを実行します。

Vue CLIでは、次のコマンドでテストを実行します。

```
  npm run test:unit
```

すると、次のように実行結果が出ます。

```
> vue-cli-service test:unit
 WEBPACK  Compiling...

  [===                       ] 10% (building)
  [========================] 98% (after emitting)

 DONE  Compiled successfully in 7740ms

  [========================] 100% (completed)
 WEBPACK  Compiled successfully in 7740ms

 MOCHA  Testing...

  HelloWorld.vue
    √ renders props.msg when passed
 1 passing (35ms)

 MOCHA  Tests completed successfully
```

これは、次の条件を検証するテストです。

- msg変数に設定したテキスト「new message」が、ブラウザー上に表示され（るVueのDOMとして含まれ）ていること

テストコードは、「`tests\unit\example.spec.js`」にあり、リスト2.1です。

リスト2.1: デフォルトのテスト例のコード

```
import { expect } from 'chai'
import { shallowMount } from '@vue/test-utils'
import HelloWorld from '@/components/HelloWorld.vue'

describe('HelloWorld.vue', () => {
  it('renders props.msg when passed', () => {
    const msg = 'new message'
    const wrapper = shallowMount(HelloWorld, {
      propsData: { msg }
    })
    expect(wrapper.text()).to.include(msg)
  })
})
```

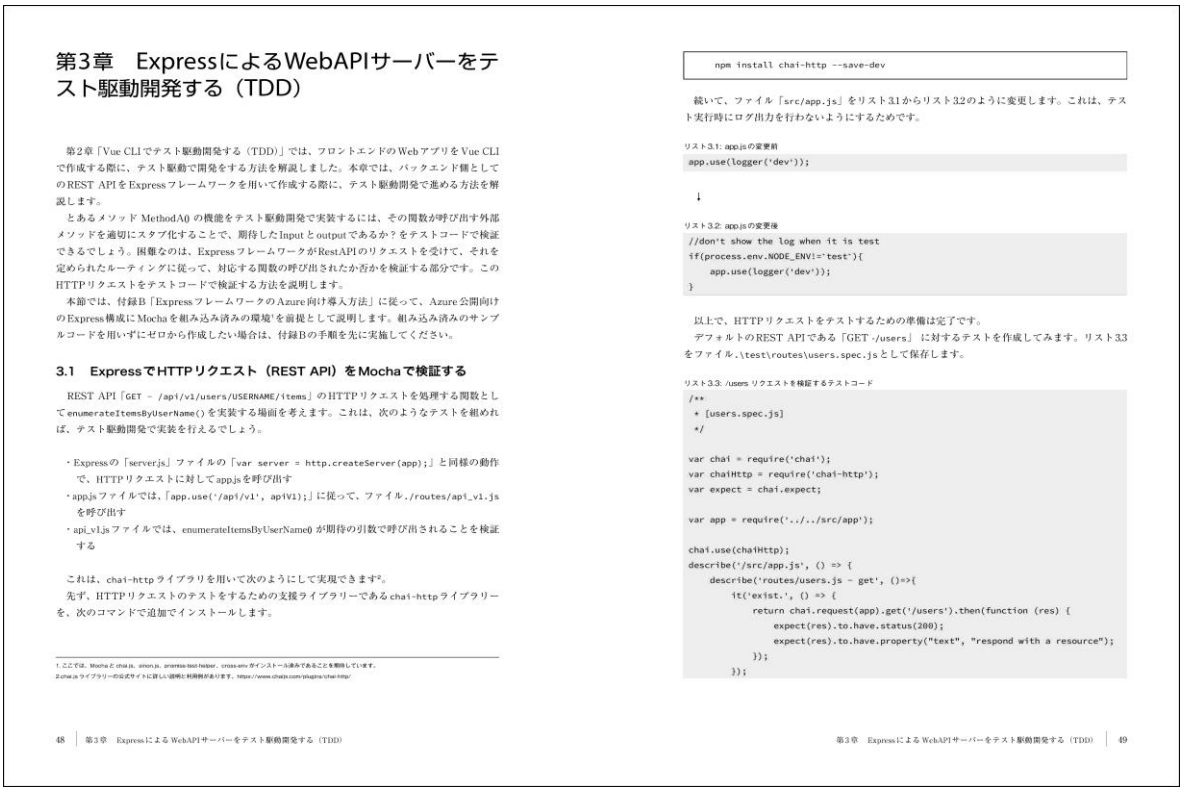
コマンド「`npm run test:unit`」は、vue-cli-serviceを経由して、Vueファイルを適切なjavascriptへトランスパイルしたのちに、テストフレームワークMochaへ渡します。Mochaは「`tests\unit\**.spec.js`」ファイルに従って、渡されたトランスパイル済みjavascriptコードをテストするように設定された上で呼び出されます[1]。

ここでは、「リスト2.1のように「`it('title', ()=>{ テスト項目 })`」と書くことで、コマンドライン「`npm run test:unit`」からテスト項目が実行される、と捉えてください。

テストコード（リスト2.1）にあるsallowMount()は、`@vue/test-utils`ライブラリが提供するテストヘルパーで、Vueファイルに基づいてVueのインスタンスを作成して返却する関数です[2]。返却されたインスタンスは、「マウントされたコンポーネントと仮想DOM、またはコンポーネントと仮想DOMをテストするメソッドを含むオブジェクト」であり、これを用いてHTMLのDOMを辿ることでVueの表示状態を検証できます。

1. この設定は、vue createコマンドを実行した際に行われます。後述しますが、babel.config.js等が設定ファイルになります。
2. https://vue-test-utils.vuejs.org/ja/api/wrapper/

Express による WebAPI サーバーをテスト駆動開発



第3章　ExpressによるWebAPIサーバーをテスト駆動開発する（TDD）

第2章「Vue CLIでテスト駆動開発する（TDD）」では、フロントエンドのWebアプリをVue CLIで作成する際に、テスト駆動で開発をする方法を解説しました。本章では、バックエンド側としてのREST APIをExpressフレームワークを用いて作成する際に、テスト駆動開発で進める方法を解説します。

とあるメソッドMethodA()の機能をテスト駆動開発で実装するには、その関数が呼び出す外部メソッドを適切にスタブ化することで、期待したInputとoutputであるか？をテストコードで検証できるでしょう。困難なのは、Expressフレームワークが RestAPIのリクエストを受けて、それを定められたルーティングに従って、対応する関数の呼び出されたか否かを検証する部分です。このHTTPリクエストをテストコードで検証する方法を説明します。

本節では、付録B「Expressフレームワークの Azure 向け導入方法」に従って、Azure公開向けのExpress構成にMochaを組み込み済みの環境[1]を前提として説明します。組み込み済みのサンプルコードを用いずにゼロから作成したい場合は、付録Bの手順を先に実施してください。

3.1　ExpressでHTTPリクエスト（REST API）をMochaで検証する

REST API「GET - /api/v1/users/USERNAME/items」のHTTPリクエストを処理する関数としてenumerateItemsByUserName()を実装する場面を考えます。これは、次のようなテストを組めれば、テスト駆動開発で実装を行えるでしょう。

- Expressの「server.js」ファイルの「var server = http.createServer(app);」と同様の動作で、HTTPリクエストに対してapp.jsを呼び出す
- app.jsファイルでは、「app.use('/api/v1', api_v1);」に従って、ファイル./routes/api_v1.jsを呼び出す
- api_v1.jsファイルでは、enumerateItemsByUserName() が期待の引数で呼び出されることを検証する

これは、chai-httpライブラリを用いて次のようにして実現できます[2]。

先ず、HTTPリクエストのテストをするための支援ライブラリーであるchai-httpライブラリーを、次のコマンドで追加でインストールします。

```
npm install chai-http --save-dev
```

続いて、ファイル「src/app.js」をリスト3.1からリスト3.2のように変更します。これは、テスト実行時にログ出力を行わないようにするためです。

リスト3.1: app.jsの変更前

```
app.use(logger('dev'));
```

↓

リスト3.2: app.jsの変更後

```
//don't show the log when it is test
if(process.env.NODE_ENV!='test'){
    app.use(logger('dev'));
}
```

以上で、HTTPリクエストをテストするための準備は完了です。

デフォルトのREST APIである「GET -/users」に対するテストを作成してみます。リスト3.3をファイル.\test\routes\users.spec.jsとして保存します。

リスト3.3: /usersリクエストを検証するテストコード

```
/**
 * [users.spec.js]
 */

var chai = require('chai');
var chaiHttp = require('chai-http');
var expect = chai.expect;

var app = require('../../src/app');

chai.use(chaiHttp);
describe('/src/app.js', () => {
    describe('routes/users.js - get', ()=>{
        it('exist.', () => {
            return chai.request(app).get('/users').then(function (res) {
                expect(res).to.have.status(200);
                expect(res).to.have.property("text", "respond with a resource");
            });
        });
```

1. ここでは、Mochaと chai.js、sinon.js、proxyquire-test-helper、cross-envがインストール済みであることを期待しています。
2. chai.jsライブラリーの公式サイトに詳しい説明と利用例があります。https://www.chaijs.com/plugins/chai-http/

＜＜目次＞＞

＜＜著者紹介＞＞

窓川ほしき

2016年にNode.jsと出会い「こんなに簡単にサーバーサイドのコードも書けるのか！」と感動し、Webブラウザーベースのツール作成を開始。「JavaScriptでの作成の手軽さとAzureでの公開の簡単さを伝えたい」と技術系同人誌の即売会に参加している。マイブームは劇場版BLAME!で、今でも観に行っている。著書に「Azure無料プランで作る！初めてのWebアプリケーション開発」「テスト駆動で作る！初めてのAzureアプリ」（いずれもインプレスR&D刊）がある。

＜＜販売ストア＞＞

電子書籍:

Amazon Kindleストア、楽天koboイーブックストア、Apple Books、紀伊國屋書店 Kinoppy、Google Play Store、honto電子書籍ストア、Sony Reader Store、BookLive!、BOOK☆WALKER

印刷書籍:

Amazon.co.jp、三省堂書店オンデマンド、honto ネットストア、楽天ブックス

※　各ストアでの販売は準備が整いしだい開始されます。

※　全国の一般書店からもご注文いただけます。

【インプレスR&D】 https://nextpublishing.jp/

株式会社インプレス R&D（本社：東京都千代田区、代表取締役社長：井芹昌信）は、デジタルファーストの次世代型電子出版プラットフォーム「NextPublishing」を運営する企業です。また自らも、NextPublishingを使った「インターネット白書」の出版などIT関連メディア事業を展開しています。

※NextPublishingは、インプレスR&Dが開発した電子出版プラットフォーム（またはメソッド）の名称です。電子書籍と印刷書籍の同時制作、プリント・オンデマンド（POD）による品切れ解消などの伝統的出版の課題を解決しています。これにより、伝統的出版では経済的に困難な多品種少部数の出版を可能にし、優秀な個人や組織が持つ多様な知の流通を目指しています。

【インプレスグループ】 https://www.impressholdings.com/

株式会社インプレスホールディングス（本社：東京都千代田区、代表取締役：唐島夏生、証券コード：東証1部9479）を持株会社とするメディアグループ。「IT」「音楽」「デザイン」「山岳・自然」「旅・鉄道」「学術・理工学」を主要テーマに専門性の高いメデイア&サービスおよびソリューション事業を展開しています。さらに、コンテンツビジネスのプラットフォーム開発・運営も手がけています。

【お問い合わせ先】

株式会社インプレスR&D NextPublishingセンター
TEL 03-6837-4820
電子メール：np-info@impress.co.jp