

2018年11月12日
株式会社インプレスR&D
<https://nextpublishing.jp/>

日曜プログラマのためのテスト駆動開発入門書！
『テスト駆動で作る！初めての Azure アプリ』発行
技術書典シリーズ、11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、『テスト駆動で作る！初めての Azure アプリ』(著者:窓川 ほしき)を発行いたします。

『テスト駆動で作る！初めての Azure アプリ』
<https://nextpublishing.jp/isbn/9784844398554>



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 1800 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 106 ページ
ISBN:978-4-8443-9855-4
発行:インプレス R&D

<< 発行主旨・内容紹介 >>

【日曜プログラマのためのテスト駆動開発入門書！】

本書は、JavaScript でテスト駆動開発を行い、実際に Azure 上に Web アプリを実装して公開してみるまでのチュートリアルガイドです。

テスト駆動開発には「テストを先に書き、あとから実装する」ことによりテストコードがそのまま設計仕様書になるメリットがあります。

日曜プログラマのようにスキマ時間でプログラミングをする際にありがちな「設計を忘れてしまってなかなか進まない」ということを避けながら、実際に Web アプリを作る方法を掲載しています。

(本書の想定読者)

- 初歩の JavaScript の知識があるプログラマ
- スキマ時間を使ってプログラミングを行いたい日曜プログラマ

(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。)

実際にライフログアプリをテスト駆動開発で行いつつ、その手法を紹介

第2章 サーバー側の機能を実装して、テストをpassさせる

本章では、前章で設計したテストコードを実際に実行しながら、機能を実装します。サンプルコードの最初の「テストの実行」は、被テスト関数数が未実装なので「失敗」します。少しずつ、被テスト関数数の実装を進めながら、テストが「成功」から「成功」へ変わっていく様子を説明します。

テストのみを実装済み（被テスト対象の関数は未実装）のサンプルコード一式（リスト2.1）をサポーターサイトからダウンロードして任意のフォルダに格納します。

リスト2.1: テストのみを実装済みのサンプルコードのサーバー側のファイルリスト

```
.gitignore
.vscode
db
LICENSE
package.json
README.md
server.js
src/api/activitylog/*
  /sql_db_io/*
  /app.js
  /public/
  /routes/
test/activitylog/*
  /sql_db_io/*
  (一部省略)
```

続いて、動作に必要なモジュールを次のコマンドで取得します。

```
npm install
```

これでテストを実行できる環境が整いました。次の節「2.1 フォルダ構造とアプリの構成概要について」では、このサンプルコードのフォルダ構造について説明します。テストの実行へ急ぎたい方は、飛ばして「2.2 テストの実行例と最初のテスト結果」へ進んでも構いません。

2.1 フォルダ構造とアプリの構成概要について

先章で記載したように、Mochaテストフレームワークは「src」と「test」のフォルダ構造が基本となります。テストコードは test フォルダ配下に、実際のアプリの機能実装コードは src フォルダ配下に格納します。

本書の目的は「テストで設計したのち、実際にアプリを実装して Azure で公開しよう」ですので、Azure での公開を想定したアプリの実装と構成になっています。リスト2.1のファイルリストのうちで、リスト2.2に示した「server.js」と「src フォルダ配下」が、Webアプリケーションの実装部分です。

リスト2.2: Expressフレームワークのファイルフォルダ構成

```
server.js      ★ Azure の Web APP のエントリーポイント。
src/app.js     ★ このファイル内に、routes/api_v1.js への呼び出しを記述。
  /api/activitylog/*
  /sql_db_io/*
  /public/*
  /routes/api_v1.js ★ このファイルにて、RESTful API と
                    api フォルダ配下の機能実装を記述する。
  /views/*
(以下略)
```

サーバー側とクライアント側（ブラウザ側）の通信には RESTful API を用います。サーバー側で実装した機能（ライフログを記録）の提供は、RESTful API を経由して提供します。RESTful API 応答と、クライアント側（ブラウザ側）向けの UI を表示する HTTP サーバーの実装は、Express フレームワークを利用します（本書でのテストの範囲からは外れます）。

なお、Express フレームワークを用いたアプリを Azure で公開する際のフォルダとファイルの構成例と、「どのようにして機能を追加するか?」についての詳細は、付録B「Express フレームワークの使い方」を参照ください。簡単に述べると、次のような手順でファイルとフォルダを構成します。

1. express-generator で Express のスケルトンを生成
2. 生成したスケルトンをリスト2.2のような構成に変更
3. test/ フォルダ配下、「ライフログを記録する機能」のテストコードを格納
4. src/api/ フォルダ配下に、「ライフログを記録する機能」を実装

これらのうち、さいごの「4.実装」以外を終えたサンプルコード一式がリスト2.1です。

Web アプリのクライアント UI も実際に設計

第4章 ライフログを記録するWebアプリのクライアント側UIを作る

本章ではクライアント側のUI側を作成していきます。GUIの描画と画面へのデータの反映を容易にするためVue.jsフレームワークを採用しますが、GUIのアクションのテストは難しい部分が多いので、本書ではテストベースでの設計は見送ります。データの加工を伴うロジック部分を対象としてテストコードから設計していきます。

本書のサンプルコードでは、クライアント側から呼び出すサーバー側の RESTful API として次のサーバー側の関数を準備することを前提とします。

- RESTful API 越しに、ユーザーの登録と削除
 - api_v1_activitylog_signup() - 新規登録
 - api_v1_activitylog_remove() - 登録削除
- RESTful API 越しに、データベースの準備
 - api_v1_activitylog_setup() - SQLite データベースのテーブルの作成
- RESTful API 越しに、ライフログデータを読み書き
 - api_v1_activitylog_show() - ユーザーのログを取得
 - api_v1_activitylog_add() - ユーザーのログを追加
 - api_v1_activitylog_delete() - ユーザーのログを削除

本章では、「RESTful API 越しに、ライフログデータを読み書き」に関する UI 周りの説明をします。「ライフログデータを読み書き」の機能は、次のように RESTful API 越しで呼び出せるものとします。

```
• /api/v1/activitylog/show - GET - api_v1_activitylog_show
• /api/v1/activitylog/add - POST - api_v1_activitylog_add
• /api/v1/activitylog/delete - POST - api_v1_activitylog_delete
```

RESTful API について

本書の範囲では、「RESTful API」は「HTTP」での通信からグラフィカルな部分を含め、データの送受信だけに特化したもの、くらいに捉えて構いません。本書のサンプルでは、次のような流れで動的にデータを取得してカレンダーに表示する設計とします。

1. 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。

1. ブラウザから、サーバー側に RESTful API を用いてデータを要求/記録する
2. サーバー側で、要求に応じたデータを作成してブラウザへ返却する。
3. ブラウザで、受け取ったデータに応じて表示を更新する。

本書で用いるサンプルコードは図4.1のような画面表示を行うように実装してあります。それぞれのボタンを押したときの動作は次のようになります。

- 「起きた」「寝る」が押されたら、ライフログをサーバー側へ記録する
- 「更新」ボタンが押されたら、ライフログの最新状態をサーバー側から取得する
- 「取り消し」ボタンが押されたら、ひとつ前のログをサーバー側から削除する

図4.1: ブラウザ側のUI



1. 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。

付録では Sinon ライブラリの API を抜粋して紹介

付録A Sinon ライブラリで良く使うAPIについて

本付録では、「Sinon ライブラリ」の中から本書において良く使う「スタブ関数の応答の設定方法」と「動作後の検証のための値の取得方法」を簡単に説明します。それぞれの設定と取得は「Sinon API」（以降 API と略記）を用いて行います。より詳細な用法やここで説明しない API については、次の Sinon の公式サイトを参照ください。

<https://sinonjs.org/>

スタブ関数の応答の設定方法は、「API Documentation - SinonJS」配下の「Stubs」のページ（次の URL です）に記載がありますので、辞書的に検索する際はこちらをお勧めします。

<https://sinonjs.org/releases/v6.1.5/stubs/>

動作後に検証するための値の取得方法は、「API Documentation - SinonJS」配下の「Spies」のページ（次の URL です）に記載がありますので、辞書的に検索する際はこちらをお勧めします。

<https://sinonjs.org/releases/v1.17.7/spies/>

「Spies」のページは、スパイ関数が持つ API の説明ですが、スタブ関数も全く同じ API を持っているため共通して使うことができます（そのため、「Stubs」のページでは記載が省略されています）。

本付録では、「var stubA = sinon.stub();」として生成したスタブ関数に対する API の使い方を説明します。

A.1 スタブ関数の動作を設定する API

スタブ関数の動作を設定する API で、本書で利用しているものは次のものです。

- ・ n 回目の呼び出し時に、値を返却する。

```
stubA.onCall(0).returns("hoge");
```

1. 呼び出された回数から数えて n 回目の呼び出し時に、指定した値を返却する。（「テストのバリエーション」も参照してください）

```
// 初期 (=0 回目) の呼び出しで "hoge" を返す。
stubA.onCall(n).returns(
  Promise.resolve({ "result": "OK" })
);
// n 回目の呼び出しで {"result": "OK"} を引数とした
// Promise.resolve() インスタンスを返す。

・引数 A で呼び出されたら、値を返却する。

stubA.withArgs("hoge").returns(
  { "message": "OK" }
);
// 引数 "hoge" で呼び出されたら、{ "message": "OK" } を
// 戻り値として返却する。

・呼び出されたら、即座に callback A 関数を引数 B を用いて実行する。

stubA.callsArgWith(0, "hoge");
// stubA( function(param1) { /* びよびよ */ } ) の呼び出しに対して、
// stubA の 0 番目 (0 オリジン) の引数を callback 関数とみなして、
// 引数 param1 に "hoge" を指定した上で即座に callback 関数を実行する。

stubA.callsArgWith(
  2,
  null, { "value": "OK" }
);
// stubA() の 2 番目 (0 オリジン) の引数を callback 関数と見なして実行する。
// 詳細は以下のコメント内を参照。

/*
  stubA( param0, param1, function(err, result){
    // 何らかのコールバック処理。
  });
  // 上記のようにスタブ関数 stubA() を呼び出した際に、
  // その 2 番目の引数 (0 オリジン) を callback 関数と捉えて、
  // function( err=null, result={"value": "OK"} );
  // として即座に実行する。
*/
```

94 | 付録A Sinon ライブラリで良く使う API について

<<目次>>

第1章 ライフログを記録する Web アプリケーションのサーバー側のテストを作成する

- 1.1 ユーザー登録機能のテストを設計する
- 1.2 Mocha とは？ Chai とは？ Sinon とは？
- 1.3 ユーザー登録機能のテストの不足分を追加する
- 1.4 ユーザー削除機能のテストを設計する（重要度に応じて Pending を利用する）

第2章 サーバー側の機能を実装して、テストを pass させる

- 2.1 フォルダ構造とアプリの構成概要について
- 2.2 テストの実行例と最初のテスト結果
- 2.3 ユーザー登録機能を実装してテストを pass させる
- 2.4 ユーザー削除機能を実装してテストを pass させる

第3章 ライブラリの I/O の実動作をテストで確認しながら実装する

- 3.1 テストフレームワークから実際の外部 I/O を試行する
- 3.2 外部 I/O をスタブ化する
- 3.3 現在時刻を内部的に利用する関数のテスト作成

第4章 ライフログを記録する Web アプリのクライアント側 UI を作る

- 4.1 関数内の時間変換のテストを作成する
- 4.2 関数内の時間変換を実装する

第5章 全体を実装して、Azure に公開する。

- 5.1 ローカルで、全体の動作確認を行う
- 5.2 Azure 上に公開して、設定と動作確認を行う
- 5.3 Azure での公開後の機能強化について

付録A Sinon ライブラリで良く使う API について

- A.1 スタブ関数の動作を設定する API
- A.2 実行後のスタブ関数の呼び出し状況を取得する API

<< 著者紹介 >>

窓川 ほしき

大学時代に、趣味で Windows アプリケーションの作成を始める。アプリは Vector で公開し、ダウンローダーのカテゴリで人気1位を獲得。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、Webブラウザベースのツール作成を開始する。「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イベントにて同人誌を頒布していたところ、商業出版の声がかかる。Web上での名前は「ほしまど」。最近のマイブームは劇場版BLAME!。

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」(インプレス R&D)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。

【株式会社インプレス 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ィア&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