OpenAPI Specification を用いた WebAPI テストの自動化

佐古田健志[1] 佐藤弓子[1] 前川智則[1]

概要：近年はインターネットを介して異なるシステムにおけるプログラムを連携させることができる Web API が増加している。しかし、Web API 公開は自社のデータやサービスを公開することであるため、セキュリティの担保やサーバへの負荷といった点で課題があり、Web API を公開する前に実施するテスト工程がソフトウェア開発において重要な工程となっている。本稿では Web API の記述方式の一つである OpenAPI Specification から網羅的な Web API のテストを自動的に作成するツールを提案する。

WebAPI test automation with OpenAPI Specification

TAKESHI SAKODA[1] YUMIKO SATO[1] TOMONORI MAEGAWA[1]

1. はじめに

アプリケーションプログラミングインターフェース（API）は、単一システム内においてプログラムとプログラムが連携することである。近年は自社で開発・運用しているサービスに外部から連携できるよう、インターネットを介して異なるシステムにおけるプログラムを連携させることができる Web API が増加している[1]。API を Web API として公開することにより、あらゆる人や企業の持つサービスと自社のサービスを連携し、自社サービス自体の価値を高めることができる。一方、Web API 公開は自社のデータやサービスを公開することであるため、セキュリティの担保やサーバへの負荷といった点で課題がある。そこで、Web API を公開する前に実施するテスト工程がソフトウェア開発において重要な工程となっている。しかし、テスト工程そのものに課題を持つ開発者も多く存在し、特にテストの抜け漏れがよくあることが課題であると認識されている[2]。また、テストの抜け漏れを抑え品質を上げるためには多くの人月が必要となるが、テストのコストは開発プロジェクトを圧迫する。

本稿では、Web API の記述方式の一つで標準仕様となっている OpenAPI Specification[3]に着目する。OpenAPI Specification では一度仕様書を作成すると、当該仕様書を元に公開用ドキュメントの作成、ソースコードの生成、およびサードパーティ製ツールを用いたテストの実行等、様々な用途に再利用できる仕様である。このため、Web API を作成する際の記述方法として OpenAPI Specification は広く使われている。我々は、OpenAPI Specification に準拠したドキュメントから抜け漏れない網羅的なテストを自動的に生成するツールを提案する。これによりテストの抜け漏れを防ぎ、開発プロジェクトにおけるテストのコストを抑制することができる。

2 章では OpenAPI Specification を用いたテスト自動化に関する関連技術について調査した結果を述べる。3 章では提案するテスト自動化ツールの設計、4 章では試作したテスト自動化ツールの評価について述べる。

2. 関連技術

Web API のテストには大きく分けて 2 つのテスト項目がある。1 つは外部に公開する Web API の仕様が正しく記述されているか否かのバリデーション、もう 1 つは外部から Web API にアクセスした際の API テストである。

1 つ目の仕様のバリデーションについては、標準仕様である OpenAPI Specification によって仕様を定めることで、OpenAPI Specification を規定している OpenAPI Initiative から提供されているバリデーションツール[4]を用いることでテストを容易に行える。

2 つ目の API についてはテスト自動化が可能と謳っているツールがいくつかある[5][6][7]。しかし、OpenAPI Initiative が提供しているサンプルファイル[8]を用いて確認したところ、OpenAPI の定義ファイルからテストを自動で実行できるツールは見つからなかった。以下では OpenAPI 定義ファイルをどの程度活用できそうかについて述べる。

2.1 Dredd

OpenAPI v2.0 であれば YAML ファイルを読み込んでテスト可能ではあるが、入力値の例とデフォルト値が設定されていないとテストできない。さらに、あらかじめファイルに記載しておいた入力値の例とデフォルト値を用いたテストしかできない。つまり、Dredd 単体では OpenAPI を読み込んだテストの自動化はできるが、網羅的なテストの実現は不可能である。

1 株式会社東芝 研究開発センター
Corporate R&D Center, Toshiba Corp.

2.2 Tavern

自身でテストコードを作成してテストするツールであり、様々なパラメータの変更を自由に記述できる。記述形式はYAMLで、ファイル命名ルールはtest_*.tarvern.yamlとなる。pytest経由で実行する。Tavern単体ではOpenAPIを読み込んで網羅的なテストを自動化することはできない。

2.3 Karate

自身でシナリオを記述して、どのAPIを利用した際にどんなレスポンスがあれば成功かを記述する。基本的には以下の3項目を設定していく。

- Given：URL
- When：method
- Then：Expect response

Givenで設定したURLパスのリソースに対して、Whenで設定したmethodやパラメータでアクセスし、その結果がThenと合致しているかをテストする。Karate単体ではOpenAPIを読み込んで網羅的なテストを自動化することはできない。

2.4 関連技術調査結果まとめ

既存ツールでは、API テストにおいて OpenAPI Specification に準拠したドキュメントから網羅的なテストを自動的に生成することができないとわかった。一方で、独自のフォーマットでテストコードを記述する必要があるが、Tavern や Karate では網羅的なテストを実行可能であることも分かった。

3. テスト自動化ツールの設計

3.1 全体設計

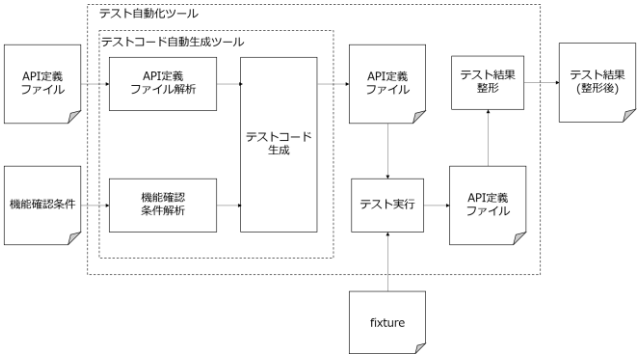

図 1：テスト自動化ツールの概要図

テスト自動化ツールの概要図を図 1 示す。テスト自動化ツールはテストコード自動生成ツールとテスト実行部の 2 つに大きく分類できる。テストコード自動生成ツールは、OpenAPI Specification によって定義された API 定義ファイルと機能確認条件を入力とし、それぞれのファイルを解析した上でテストコードを出力する。テスト実行部は、テストコード自動生成ツールが生成したテストコードを読み込み、2 章で述べた Tavern や Karate といった OSS ツールを活用してテストを実施する。

3.2 詳細設計

本稿における機能確認条件とは、Web API テストの各 API に対するテストパターンやそのテスト成否のパラメータ条件等を人間可読・機械可読のフォーマットで記したファイルのことである。機能確認条件の例として、Web API におけるパラメータのタイプが integer 型である場合に境界値テストを実施するという内容を JSON で記述することが挙げられる。

また、本稿におけるテストコード生成部は、解析した OpenAPI Specification の構造データを元に、各パスにおけるパラメータ情報を順に確認する。OpenAPI Specification には各 API において Example の形で API パラメータの典型的なパラメータ設定情報がある。提案するテストコード自動生成ツールでは Example を参考に正常系のテストコードを生成する。この際、当該パラメータが機能確認条件で定義されているパラメータに対する条件と合致する場合、機能確認条件の内容に従ってパラメータ値を変更し、正常系あるいは異常系のテストコードを生成する。例として、API パラメータのタイプが integer 型である場合、min や max が設定されていると当該設定値を用いた境界値テストを生成する。なお、min や max が定義されていない場合は OpenAPI Specification における各パラメータタイプの閾値を参照して独自に min と max を設定した上で境界値テストを生成する。

パスパラメータを含むパスに対する操作は、Web API を搭載するサービス提供サーバに存在する情報が必要となる。OpenAPI Specification に準拠したドキュメントのみを参考にしてテストコードを生成する場合、テスト対象のサーバにしか存在しない情報を利用できない。そこで、テストコード自動生成ツールとは別に新たに外部ファイル fixture を用意する。テストコード自動生成ツールは生成するテストコードに加え、自動生成したテストコードだけでは取得できない情報を補完する情報（これを fixture と呼ぶ）を用いてテストを実行する。なお、fixture の実装方法には 2 通りの考え方があり、パスパラメータに相当する値を直接書き換えたテストコードを出力する場合と、外部関数を読み出すテストコードを出力する場合である。それぞれについて以降で述べる。

(1) 置換する値を直接記述する場合

fixture の記述例を以下に示す。

```
petId:
- 1234567890
```

この例では、petId というパスパラメータがあった場合に置換する値を記述している。これをテストコード自動生成ツールが読み込み、出力するテストコード例を以下に示す。

```
---
test_name: GET_/pet/{petId}
stages:
- name: GET_/pet/{petId}
  request:
    url: 'https://petstore3.swagger.io/api/v3/pet/1234567890'
```

```
    method: GET
  response:
    status_code: 200
```

(2) **外部関数から値を動的に取得する場合**

fixture の記述例を以下に示す。

```
petId:
- call:
    fixture.create_pet
  extra_kwargs:
    name: bingo
  save:
    petId: id
```

この例では、petId というパスパラメータがあった場合に、fixtture.py 内の関数 create_pet に引数 name の値を渡し、返り値の id を petId に代入するという記述がされている。これをテストコード自動生成ツールが読み込み、出力するテストコード例と fixture.py 例を以下に示す。

```
---
test_name: GET_/pet/{petId}
stages:
- name: GET_/pet/{petId}
  request:
    function: fixture:create_pet
    extra_kwards:
      name: bingo
    url: 'https://petstore3.swagger.io/api/v3/pet/{petId}'
    method: GET
  response:
    status_code: 200
```

```
import requests
from box import Box

def create_pet(name):
    response =
requests.post('https://petstore3.swagger.io/api/v3/pet',
data={'name': '{name}'})
    id = {
            "id": response.json()["id"]
    }
    return Box(id)
```

3.3 実装評価向けの仕様と前提条件

実装評価向けのテストコード自動生成ツールの仕様は表 1 の通りである

表 1：テストコード自動生成ツールの仕様

OpenAPI Specification のバージョン	3
対象とするメソッド	POST
ステータスコード	200、201、400
レスポンスボディ内のパラメータ設定方法	リクエスト内容と同値
認証 API の取り扱い	自動生成対象外

OpenAPI Specification には複数のバージョンがある。バージョン 2 にあったオブジェクトがバージョン 3 では components に統合/廃止される、バージョン 3 になって host を複数指定できるようになるなど、バージョン 2 と 3 では公報互換性のない仕様変更が行われている。今回はポイントを絞って実装評価を行うため、バージョン 3 のみを対象とする。

Web API で利用される HTTP メソッドは GET、POST、PUT、DELETE などがあるが、GET、PUT、DELETE などのメソッドでのテストではサーバに登録済みの情報つまり OpenAPI Specification に準拠したドキュメント以外の情報が必要である。今回はポイントを絞って実装評価を行うため、新たに情報を登録する操作である POST のみを対象とする。

Web API のリクエストに対するレスポンスには、情報レスポンスの 100 番台、成功レスポンスの 200 番台、リダイレクトの 300 番台、クライアントエラーの 400 番台、サーバエラーの 500 番台といった多種のステータスコードが付随する。200 または 201（リクエストの成功）と 400（リクエストの失敗）に対応する。

レスポンスに含まれるデータの検証では、送信したリクエストボディのプロパティ名と同じプロパティ名がレスポンスボディにも含まれる場合は、リクエストボディに使用した値が返ってくるものとして値を判定する。レスポンスボディで初めて登場するプロパティに関してはタイプのみ確認する。

テストコード自動生成ツールは API 単体のテストコードを自動生成する。個々のテストは独立しており、順序は関係ない。一方で、多くの Web API で必須となる認証処理は API を適切な順序で操作する必要がある。今回の実装評価では、OpenAPI Specification から読み取れる情報のみでテストコードを自動生成することを目指すため、OpenAPI Specification では表現できない API の利用順序が必要となる認証部分については自動生成対象外とする。

4. 試作したテスト自動化ツールの評価結果と考察

本章では fixture に関する内容以外について試作実装したテストコード自動生成ツールにおいて、どこまでテストコードを自動生成できるかを確認する。なお、テスト実行部としては Tavern を用い、テスト自動化ツールにおけるテスト対象は、東芝が提供している IoT 基盤サービス HABANEROTS®[9]の WebAPI とする。動作環境を図 2 に示す。

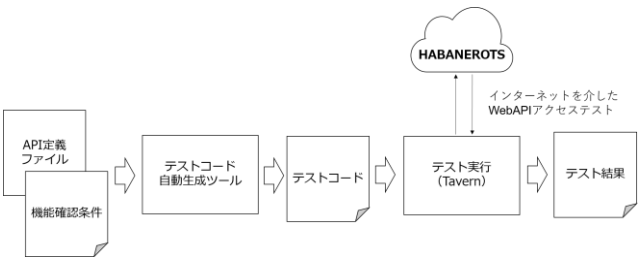

図 2：試作実装したシステムの動作環境

テストコード自動生成ツールの出力の一部を以下に示す。

```
---
includes:
  - !include common.yaml
strict:
  - json:off
```

```
test_name: POST_/users
stages:
- *auth
- *self
- *login
- *grant
- *token
- name: POST_/users
  request:
    url: '{protocol:s}://{apihost:s}/users'
    method: POST
    headers:
      Authorization: Bearer {access_token}
    json:
      email: john.smith@example.com
      lang: ja
      name: John Smith
      tags:
        foo: null
  response:
    status_code: 201

---
includes:
  - !include common.yaml
strict:
  - json:off
test_name: POST_/users_email_lost
stages:
- *auth
- *self
- *login
- *grant
- *token
- name: POST_/users_email_lost
  request:
    url: '{protocol:s}://{apihost:s}/users'
    method: POST
    headers:
      Authorization: Bearer {access_token}
    json:
      lang: ja
      name: John Smith
      tags:
        foo: null
  response:
    status_code: 400

---
includes:
  - !include common.yaml
strict:
  - json:off
test_name: POST_/users_email_null
stages:
- *auth
- *self
- *login
- *grant
- *token
- name: POST_/users_email_null
  request:
    url: '{protocol:s}://{apihost:s}/users'
    method: POST
    headers:
      Authorization: Bearer {access_token}
    json:
      email: null
      lang: ja
      name: John Smith
      tags:
        foo: null
  response:
    status_code: 400

---
includes:
  - !include common.yaml
strict:
  - json:off
test_name: POST_/users_email_maxLen
stages:
- *auth
- *self
- *login
```

```
- *grant
- *token
- name: POST_/users_email_maxLen
  request:
    url: '{protocol:s}://{apihost:s}/users'
    method: POST
    headers:
      Authorization: Bearer {access_token}
    json:
      email: 
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaa@example.com
      lang: ja
      name: John Smith
      tags:
        foo: null
  response:
    status_code: 201

---
includes:
  - !include common.yaml
strict:
  - json:off
test_name: POST_/users_email_maxLen_over
stages:
- *auth
- *self
- *login
- *grant
- *token
- name: POST_/users_email_maxLen_over
  request:
    url: '{protocol:s}://{apihost:s}/users'
    method: POST
    headers:
      Authorization: Bearer {access_token}
    json:
      email: 
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaa@example.com
      lang: ja
      name: John Smith
      tags:
        foo: null
  response:
    status_code: 400

---
includes:
  - !include common.yaml
strict:
  - json:off
test_name: POST_/users_email_empty
stages:
- *auth
- *self
- *login
- *grant
- *token
- name: POST_/users_email_empty
  request:
    url: '{protocol:s}://{apihost:s}/users'
    method: POST
    headers:
      Authorization: Bearer {access_token}
    json:
      email: ""
      lang: ja
      name: John Smith
      tags:
        foo: null
  response:
    status_code: 400
```

自動生成されたテストコード必須プロパティの欠損テストとプロパティ値の NULL テストと境界値テストが生成されていることが分かる（上記網掛部）。

4.1 テストの実行結果

テストの実行結果のサマリの一部を以下に示す。

```
=================================== test session starts
===================================
platform linux -- Python 3.10.1, pytest-6.2.5, py-1.11.0,
pluggy-1.0.0 --
~/.anyenv/envs/pyenv/versions/3.10.1/bin/python
cachedir: .pytest_cache
rootdir: ~/jolokia
plugins: tavern-1.18.0
collected 63 items

test_token_habanero.tavern.yaml::HABANEROTS PASSED
                    [  1%]
. . .
test_token_habanero.tavern.yaml::POST_/users PASSED
                    [ 46%]
test_token_habanero.tavern.yaml::POST_/users_email_lost
PASSED              [ 47%]
test_token_habanero.tavern.yaml::POST_/users_email_null
PASSED              [ 49%]
test_token_habanero.tavern.yaml::POST_/users_email_maxLen
FAILED              [ 50%]
test_token_habanero.tavern.yaml::POST_/users_email_maxLen_ov
er PASSED           [ 52%]
test_token_habanero.tavern.yaml::POST_/users_email_empty
PASSED              [ 53%]
test_token_habanero.tavern.yaml::POST_/users_lang_null
PASSED              [ 55%]
test_token_habanero.tavern.yaml::POST_/users_lang_empty
FAILED              [ 57%]
test_token_habanero.tavern.yaml::POST_/users_name_lost
PASSED              [ 58%]
test_token_habanero.tavern.yaml::POST_/users_name_null
PASSED              [ 60%]
test_token_habanero.tavern.yaml::POST_/users_name_empty
PASSED              [ 61%]
test_token_habanero.tavern.yaml::POST_/users_tags_foo_nullVa
lue PASSED          [ 63%]
test_token_habanero.tavern.yaml::POST_/users_tags_null
PASSED              [ 65%]
test_token_habanero.tavern.yaml::POST_/reset_password  PASSED
                    [ 66%]
test_token_habanero.tavern.yaml::POST_/reset_password_email_
lost PASSED                    [ 68%]
test_token_habanero.tavern.yaml::POST_/reset_password_email_
null PASSED                    [ 69%]
test_token_habanero.tavern.yaml::POST_/reset_password_email_
maxLen FAILED                  [ 71%]
test_token_habanero.tavern.yaml::POST_/reset_password_email_
maxLen_over PASSED    [ 73%]
test_token_habanero.tavern.yaml::POST_/reset_password_email_
empty PASSED                   [ 74%]
. . .
```

自動生成されたテストの合計数は 63 であった。また、元々人力で作成していた「POST メソッドをもつパスパラメータを含まないパス」に関連するテスト項目数は 194 であり、約 3 割のテストコードを自動生成できたと言える。

4.2 実行結果に対する考察

自動生成したテストコードが約 3 割にとどまった原因について考察する。

まず、他の API を利用したデータ取得が必要なテストへの対応ができていない。HABANEROTS®の Web API のパスの一部には、別 API でリソースを作成した際に知ることができる情報が必要なプロパティがある。このプロパティに対して、正常系テスト、空／NULL／未登録／アクセス権限がない等の異常系を生成する必要がある。しかし、今回の試作ではこの様なプロパティについてスコープ外としていたため、評価対象のサービスにおいては、自動生成したテストコード数が低下したと考えられる。

また、権限を変更した場合のテストに対応できていない。つまり、送信データはそのままで、API を実行するユーザの権限を変更しながらその応答を確認するテストがある。しかし、本稿で提案するテスト自動化ツールでは OpenAPI Specification の情報と機能確認条件から自動的にテストを生成しているが、権限の取り扱いについては OpenAPI Specification では表現することができない。このため、試作したテストコード自動生成ツールではテストを生成できず、自動生成したテストコード数が低下したと考えられる。

別 API のリソース情報を必要とするプロパティについては、fixture を活用することで自動生成したテストコード数の向上が見込める。対して、権限の変更についての対応は今後の課題である。

5. おわりに

本稿では OpenAPI Specification によって定義された定義ファイルを用いた Web API のテスト自動化ツールの設計、および試作実装での動作状況の確認を行なった。今回評価に用いた Web API に対して自動生成できたテストコードは約 3 割であった。一方で、今回の試作では実装できなかった fixture を実装することで更なる自動生成したテストコード数の向上が見込めることも分かった。今後は fixture の詳細設計および試作実装を行い、テスト自動生成ツールにおける自動生成可能なテストコード数の更なる向上を目指す。

参考文献

[1] Programmable Web HP, https://www.programmableweb.com/news/apis-show-faster-growth-rate-2019-previous-years/research/2019/07/17

[2] バルテス株式会社，第三者検証に関するアンケート調査結果を公開，https://www.valtes.co.jp/news/2021/202104072917

[3] OpenAPI Specification, https://spec.openapis.org/oas/latest.html

[4] OpenAPI.Tools, https://openapi.tools/#data-validators

[5] Dredd, https://dredd.org/en/latest/

[6] Tavern, https://tavern.readthedocs.io/en/latest/

[7] Karate, https://github.com/karatelabs/karate

[8] Example, OpenAPI Initiative，入手先 https://github.com/OAI/OpenAPI-Specification/blob/master/examples/v3.0/petstore.yaml.

[9] 内田，樋口，CPS サービスの迅速な立ち上げに貢献する 東芝 IoT 基盤サービス HABANEROTS における サービスメッシュの活用，https://www.global.toshiba/content/dam/toshiba/migration/corp/techReviewAssets/tech/review/2020/05/75_05pdf/a09.pdf