



無料電子ブック

学習

microservices

Free unaffiliated eBook created from
Stack Overflow contributors.

#microservi

ces

.....	1
1:	2
.....	2
Examples.....	2
Microservices.....	2
API.....	2
API.....	6
2: API	14
.....	14
Examples.....	14
.....	14
.....	16

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: [microservices](#)

It is an unofficial and free microservices ebook created for educational purposes. All the content is extracted from [Stack Overflow Documentation](#), which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official microservices.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

1: マイクロサービスの

このセクションでは、マイクロサービスのと、がそれをするについてします。

また、マイクロサービスのきなテーマについてもし、するトピックにリンクするがあります。マイクロサービスのドキュメンテーションはしいものなので、それらのトピックのバージョンをするがあります。

Examples

Microservices プラットフォームのためのなチェックリスト

- CI / CDパイプライン
- とサービス
- APIドキュメント
- APIゲートウェイ
- ログツールの
- サービスモニタ
- インフラストラクチャの
- サーバー

APIドキュメント

Spring REST Docsをして、サービスをしてください。これは、サービスロジックがにドキュメントとインラインであることをするなフレームワークです。これをうには、サービスのテストをするがあります。

ドキュメンテーションとサービスののにがある、テストはします。

に、Mavenプロジェクトのドキュメントをするためのサンプルのをします。

あなたのpomにこのをしてください

```
<dependency>
  <groupId>org.springframework.data</groupId>
  <artifactId>spring-data-rest-webmvc</artifactId>
  <version>2.6.6.RELEASE</version>
</dependency>
```

build.pluginsタグのにasciidocプラグインをします

```
<plugin>
  <groupId>org.asciidoctor</groupId>
  <artifactId>asciidoctor-maven-plugin</artifactId>
  <version>1.5.3</version>
  <executions>
```

```

        <execution>
            <id>generate-docs</id>
            <phase>prepare-package</phase>
            <goals>
                <goal>process-asciidoc</goal>
            </goals>
            <configuration>
                <backend>html</backend>
                <doctype>book</doctype>
            </configuration>
        </execution>
    </executions>
    <dependencies>
        <dependency>
            <groupId>org.springframework.restdocs</groupId>
            <artifactId>spring-restdocs-asciidoctor</artifactId>
            <version>1.2.0.RELEASE</version>
        </dependency>
    </dependencies>
</plugin>

```

に、したいコントローラのサンプルをてみましょう

```

package com.hospital.user.service.controller;

import org.springframework.hateoas.Resource;
import org.springframework.http.HttpStatus;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

import com.hospital.user.service.entity.User;
import com.hospital.user.service.exception.UserNotFoundException;
import com.hospital.user.service.repository.UserCrudRepository;
import com.hospital.user.service.resource.assembler.UserResourceAssembler;

@Controller
@RequestMapping("/api/user")
public class SampleController {

    final UserCrudRepository userRepository;

    final UserResourceAssembler userResourceAssembler;

    final BCryptPasswordEncoder passwordEncoder;

    public SampleController(UserCrudRepository userCrudRepository, UserResourceAssembler
userResourceAssembler, BCryptPasswordEncoder passwordEncoder){
        this.userRepository = userCrudRepository;
        this.userResourceAssembler = userResourceAssembler;
        this.passwordEncoder = passwordEncoder;
    }

    @RequestMapping(method = RequestMethod.GET, value =("/{userId}", produces = {

```

```

MediaType.APPLICATION_JSON_VALUE))
ResponseEntity<Resource<User>> getUser(@PathVariable String userId){
    User user = (User) this.userRepository.findOne(userId);
    if(user==null){
        throw new UserNotFoundException("No record found for userid"+ userId);
    }
    Resource<User> resource = this.userResourceAssembler.toResource(user);
    return new ResponseEntity<Resource<User>>(resource, HttpStatus.OK);
}
}

```

に、サービスのテストケースをします。

```

package com.hospital.user.service;

import org.junit.Before;
import org.junit.Rule;
import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.restdocs.JUnitRestDocumentation;
import org.springframework.restdocs.mockmvc.RestDocumentationResultHandler;
import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;
import org.springframework.test.context.web.WebAppConfiguration;
import org.springframework.test.web.servlet.MockMvc;
import org.springframework.test.web.servlet.setup.MockMvcBuilders;
import org.springframework.web.context.WebApplicationContext;
import org.springframework.web.servlet.config.annotation.EnableWebMvc;

import static org.springframework.restdocs.operation.preprocess.Preprocessors.prettyPrint;
import static
org.springframework.restdocs.operation.preprocess.Preprocessors.preprocessRequest;

```

import static

org.springframework.restdocs.operation.preprocess.Preprocessors.preprocessResponse;

```

import static org.springframework.restdocs.mockmvc.MockMvcRestDocumentation.document;
import static
org.springframework.restdocs.mockmvc.MockMvcRestDocumentation.documentationConfiguration;

import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.status;

import static org.springframework.restdocs.mockmvc.RestDocumentationRequestBuilders.get;

import static org.springframework.restdocs.headers.HeaderDocumentation.headerWithName;
import static org.springframework.restdocs.headers.HeaderDocumentation.responseHeaders;

import static org.springframework.restdocs.payload.PayloadDocumentation.fieldWithPath;
import static org.springframework.restdocs.payload.PayloadDocumentation.responseFields;

@RunWith(SpringJUnit4ClassRunner.class)
@WebAppConfiguration
@EnableWebMvc
@ComponentScan( basePackages = { "com.hospital.user.service" } )
@SpringBootTest

```

```

public class SampleControllerTest {

private RestDocumentationResultHandler documentationHandler;

@Rule public final JUnitRestDocumentation restDocumentation = new
JUnitRestDocumentation("target/generated-snippets");

@Autowired private WebApplicationContext context;
private MockMvc mockMvc;

@Before
public void setUp(){

    this.documentationHandler = document("{method-name}", //this will create files with
the test method name
        preprocessRequest(prettyPrint()), // to print the request
        preprocessResponse(prettyPrint()));

    this.mockMvc = MockMvcBuilders.webAppContextSetup(this.context)
        .apply(documentationConfiguration(this.restDocumentation)
            .uris()
            //.withScheme("https") Specify this for https
            .withHost("recruitforceuserservice") //Define the host name
            .withPort(8443)
        )
        .alwaysDo(this.documentationHandler)
        .build();
}

@Test
public void getUser() throws Exception {
    // tag::links[]

this.mockMvc.perform(get("/api/user/"+ "591310c3d5eb3a37183ab0d3").header("Authorization",
    "Bearer
eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJiaGFyZGhhai5uaXRpc2gxOSIsInJvbGVzIjpbIkFETUl0IiwiaWF0Ij0sImZlcyI6Imh0dHA6Ly9ob2
.

        .andExpect(status().isOk())
        .andDo(this.documentationHandler.document(

            responseHeaders(
                headerWithName("Content-Type").description("The Content-Type
of the payload: `application/json`") // Asserts that the response should have this header.
            ),

            responseFields(
                fieldWithPath("username").description("Unique name for the
record"), // Asserts that the response should have this field
                fieldWithPath("password").description("password of the user"),
                fieldWithPath("securityAnswer").description("Security answer
which would be used to validate the user while the password is reset."),
                fieldWithPath("securityQuestion").description("Security
question to reset the password"),
                fieldWithPath("email").description("Email of the user"),
                fieldWithPath("roles").description("Assigned roles of the
user"),
                fieldWithPath("id").description("Unique identifier of an
user"),
                fieldWithPath("_links").ignored()
            )
        ));
}

```

```
}
```

```
}
```

については、このリファレンスにしてください <http://docs.spring.io/spring-restdocs/docs/current/reference/html5/>

APIドキュメントのサンプル

Spring REST Docsをして、サービスをしてください。これは、サービスロジックがにドキュメントとインラインであることをするなフレームワークです。これをうには、サービスのテストをするがあります。

ドキュメンテーションとサービスのにがある、テストはします。

に、Mavenプロジェクトでドキュメントをするためのサンプルのをします。

このをpomファイルにします。

```
<dependency>
  <groupId>org.springframework.data</groupId>
  <artifactId>spring-data-rest-webmvc</artifactId>
  <version>2.6.6.RELEASE</version>
</dependency>
```

また、ASCIIドキュメントプラグインをして、build.puginタグのにドキュメントをします

```
<plugin>
  <groupId>org.asciidoctor</groupId>
  <artifactId>asciidoctor-maven-plugin</artifactId>
  <version>1.5.3</version>
  <executions>
    <execution>
      <id>generate-docs</id>
      <phase>prepare-package</phase>
      <goals>
        <goal>process-asciidoc</goal>
      </goals>
      <configuration>
        <backend>html</backend>
        <doctype>book</doctype>
      </configuration>
    </execution>
  </executions>
  <dependencies>
    <dependency>
      <groupId>org.springframework.restdocs</groupId>
      <artifactId>spring-restdocs-asciidoctor</artifactId>
      <version>1.2.0.RELEASE</version>
    </dependency>
  </dependencies>
</plugin>
```

さて、サンプルとして、したいコントローラサービスをしてみましょう。

```
package com.hospital.user.service.controller;

import org.springframework.hateoas.Resource;
import org.springframework.http.HttpStatus;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

import com.hospital.user.service.entity.User;
import com.hospital.user.service.exception.UserNotFoundException;
import com.hospital.user.service.repository.UserCrudRepository;
import com.hospital.user.service.resource.assembler.UserResourceAssembler;

@Controller
@RequestMapping("/api/user")
public class SampleController {

    final UserCrudRepository userRepository;

    final UserResourceAssembler userResourceAssembler;

    final BCryptPasswordEncoder passwordEncoder;

    public SampleController(UserCrudRepository userCrudRepository, UserResourceAssembler
userResourceAssembler, BCryptPasswordEncoder passwordEncoder){
        this.userRepository = userCrudRepository;
        this.userResourceAssembler = userResourceAssembler;
        this.passwordEncoder = passwordEncoder;
    }

    @RequestMapping(method = RequestMethod.GET, value =("/{userId}", produces = {
MediaType.APPLICATION_JSON_VALUE}))
    ResponseEntity<Resource<User>> getUser(@PathVariable String userId){
        User user = (User) this.userRepository.findOne(userId);
        if(user==null){
            throw new UserNotFoundException("No record found for userid"+ userId);
        }
        Resource<User> resource = this.userResourceAssembler.toResource(user);
        return new ResponseEntity<Resource<User>>(resource, HttpStatus.OK);
    }

}
```

このサービスをテストするテストケースをしましょう

```
package com.hospital.user.service;

import org.junit.Before;
import org.junit.Rule;
import org.junit.Test;
import org.junit.runner.RunWith;
```

```

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.restdocs.JUnitRestDocumentation;
import org.springframework.restdocs.mockmvc.RestDocumentationResultHandler;
import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;
import org.springframework.test.context.web.WebAppConfiguration;
import org.springframework.test.web.servlet.MockMvc;
import org.springframework.test.web.servlet.setup.MockMvcBuilders;
import org.springframework.web.context.WebApplicationContext;
import org.springframework.web.servlet.config.annotation.EnableWebMvc;

import static org.springframework.restdocs.operation.preprocess.Preprocessors.prettyPrint;
import static
org.springframework.restdocs.operation.preprocess.Preprocessors.preprocessRequest;
import static
org.springframework.restdocs.operation.preprocess.Preprocessors.preprocessResponse;

import static org.springframework.restdocs.mockmvc.MockMvcRestDocumentation.document;
import static
org.springframework.restdocs.mockmvc.MockMvcRestDocumentation.documentationConfiguration;

import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.status;

import static org.springframework.restdocs.mockmvc.RestDocumentationRequestBuilders.get;

import static org.springframework.restdocs.headers.HeaderDocumentation.headerWithName;
import static org.springframework.restdocs.headers.HeaderDocumentation.responseHeaders;

import static org.springframework.restdocs.payload.PayloadDocumentation.fieldWithPath;
import static org.springframework.restdocs.payload.PayloadDocumentation.responseFields;

@RunWith(SpringJUnit4ClassRunner.class)
@WebAppConfiguration
@EnableWebMvc
@ComponentScan( basePackages = { "com.hospital.user.service" } )
@SpringBootTest
public class SampleControllerTest {

private RestDocumentationResultHandler documentationHandler;

@Rule public final JUnitRestDocumentation restDocumentation = new
JUnitRestDocumentation("target/generated-snippets");

@Autowired private WebApplicationContext context;
private MockMvc mockMvc;

@Before
public void setUp(){

    this.documentationHandler = document("{method-name}", //Documents would be generated
by the test method name.
        preprocessRequest(prettyPrint()), //To print request
        preprocessResponse(prettyPrint()));

    this.mockMvc = MockMvcBuilders.webAppContextSetup(this.context)
        .apply(documentationConfiguration(this.restDocumentation)
            .uris()
            //withScheme("https") Specify this for https
            .withHost("recruitforceuserservice") //To use the hostname

```

[illegible]

テストをすると、いくつかのファイルがターゲットフォルダのにされます。

```
> 📁 src
  ▾ 📁 target
    > 📁 generated-docs
    ▾ 📁 generated-snippets
      ▾ 📁 get-user
        📄 curl-request.adoc
        📄 http-request.adoc
        📄 http-response.adoc
        📄 httpie-request.adoc
        📄 response-fields.adoc
        📄 response-headers.adoc
```

ソースフォルダを**src / main / asciidocs**としてし、**adoc**の doc ファイルをして、サービスのをし
ます。サンプルのdocファイル

```
[[resources]]
= Resources

User: Have the information about an User. It has following fields:

include::{snippets}/get-user/response-fields.adoc[]

[[resources-user]]
== User

The User resources has all the information about the application user. This resource
is being used to perform all CRUD operations on User entity.

[[resources-user-retrieve]]
=== Retrieve User

A `GET` request gets a User.

operation::get-user[snippets='response-fields,curl-request,http-response']
```

docファイルの**include** タグはスニペットをめることです。ドキュメントをするをすることができます。

すべてのがったら、 **Maven Build**をします。テストがされ、asciidocプラグインは
target.generate-docsフォルダのにドキュメントhtmlファイルをしします。されるファイルはのよ
うになります。

Table of Contents

Overview

HTTP verbs

HTTP status codes

Headers

Hypermedia

Resources

User

Retrieve User

Response fields

Example request

Example response

Links

Retrieve User

A `GET` request gets a

Response fields

Path
username
password
securityAnswer
securityQuestion
email
roles
id

Table of Contents

Overview

HTTP verbs

HTTP status codes

Headers

Hypermedia

Resources

User

Retrieve User

Response fields

Example request

Example response

Links

Example request

```
$ curl 'http://riptutorial.com/api/users/1' \
  -H 'Authorization: Basic eyJhbGciOiJIUzUxMiIsInR5cCI6IkpXZXQh0dHA6Ly9ob3NwaXQhIHR5cGU6ImF1dG8iLCJ1aWQiOiJhdXNjaaodsIzKX4y' --data '{"username": "bha", "password": "1234567890"}'
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 256
```

```
{
  "password" : "1234567890",
  "securityQuestion" : "What is your favorite color?",
  "securityAnswer" : "Blue",
  "id" : "591310",
  "username" : "bha",
  "email" : "bha@riptutorial.com",
  "roles" : [ "ADMIN", "USER" ],
  "_links" : {
    "self" : {
      "href" : "http://riptutorial.com/api/users/1"
    }
  }
}
```

```
"self" : {  
  "href" :  
}  
}  
}
```

あなたのMavenビルドがされるたびに、あなたのドキュメントがされ、にあなたのサービスとインラインになるでしょう。このドキュメントをサービスのコンシューマ/クライアントにするだけです。

ハッピーコーディング。

オンラインでマイクロサービスのをむ <https://riptutorial.com/ja/microservices/topic/7719/マイクロサービスの>

2: APIゲートウェイ

き

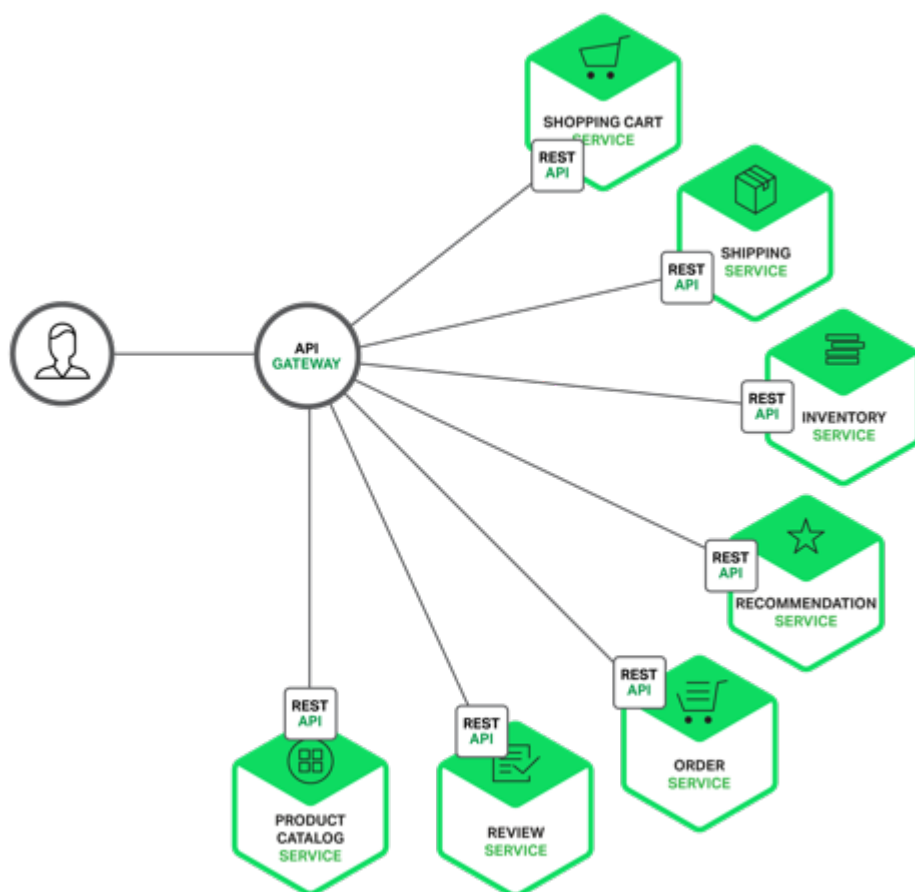
マイクロサービスアーキテクチャは、アプリケーションをりしてしたアプリケーションをするためのきなをします。マイクロサービスは、にしてテストででなければなりません。

しかし、あなたはあまりにもくのサービスをしけるので、APIゲートウェイをつがあります。

すべてのサービスをクライアントにすることはできません。すべてのマイクロサービスのゲートキーパーとしてするレイヤーがです。あなたのすべてのサービスのエントリーポイント。

Examples

ショッピングカートサービス、サービス、サービスなど、さまざまなマイクロサービスをつEコマースクラウドがあるとします。のへのエッジサービスとしてAPIゲートウェイがです。



APIゲートウェイは、マイクロサービスにするホストとポートをします。、すべてのクライアントは、APIゲートウェイである1つのサーバーURLをるがあります。のMiroserviceのは、あなたのクライアントアプリケーションにをとしません。APIゲートウェイはをするたびに、をのマイク

ロサービスにルーティングします。

オンラインでAPIゲートウェイをむ <https://riptutorial.com/ja/microservices/topic/10904/apiゲートウェイ>

クレジット

S. No		Contributors
1	マイクロサービスの	Community , Dinusha , mohan08p , Nitish Bhardwaj
2	APIゲートウェイ	Nitish Bhardwaj