Бесплатная электронная книга

УЧУСЬ

microservices

Free unaffiliated eBook created from
Stack Overflow contributors.

#microservi

ces

Около

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.

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

глава 1: Начало работы с микросервисами

замечания

В этом разделе представлен обзор того, какие микросервисы есть, и почему разработчик может захотеть его использовать.

Следует также упомянуть о любых крупных предметах в рамках микросервисов и ссылки на соответствующие темы. Поскольку документация для микросервисов новая, вам может потребоваться создать начальные версии этих связанных тем.

Examples

Основной контрольный список для платформы Microservices

- Конвейер CI / CD
- Централизованная служба проверки подлинности и авторизации
- Документация API
- Шлюз API
- Централизовать инструмент управления журналом
- Сервисный монитор
- Автоматизация инфраструктуры
- Централизованный конфигурационный сервер

Документация API

Используйте документы **Spring REST** для документирования ваших услуг. Это мощная структура, которая гарантирует, что логика Service всегда соответствует документации. Для этого вам нужно будет написать интеграционные тесты для ваших услуг.

Если есть какие-либо несоответствия в поведении документации и обслуживания, тесты потерпят неудачу.

Ниже приведен пример примера создания документов для проекта maven.

Добавьте эту зависимость в свой pom:

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

и добавьте плагин asciidoc под тегом build.plugins:

```
<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>
```

Теперь давайте возьмем образец контроллера, который мы хотим документировать:

```
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);
 }
}
```

Теперь напишите тестовый пример для службы:

```
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.eyJzdWIiOiJiaGFyZHdhai5uaXRpc2gxOSIsInJvbGVzIjpbIkFETUlOIl0sImlzcyI6Imh0dHA6Ly9ob

                .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()
                        )
                        ));
    }
```

}

Пожалуйста, следуйте этой ссылке для получения дополнительной информации: http://docs.spring.io/spring-restdocs/docs/current/reference/html5/

Пример документации API

Используйте документы **Spring REST** для документирования ваших услуг. Это мощная структура, которая гарантирует, что логика Service всегда соответствует документации. Для этого вам нужно будет написать интеграционные тесты для ваших услуг.

Если есть какие-либо несоответствия в поведении документации и обслуживания, тесты потерпят неудачу.

Ниже приведен пример примера создания документов в проекте maven:

Добавьте эту зависимость в файл pom:

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

Кроме того, добавьте плагин ASCII docs для создания документов в теге build.puglin

```
<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>
```

Теперь, в качестве образца, давайте создадим службу контроллера, которую хотим документировать.

```
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);
}
```

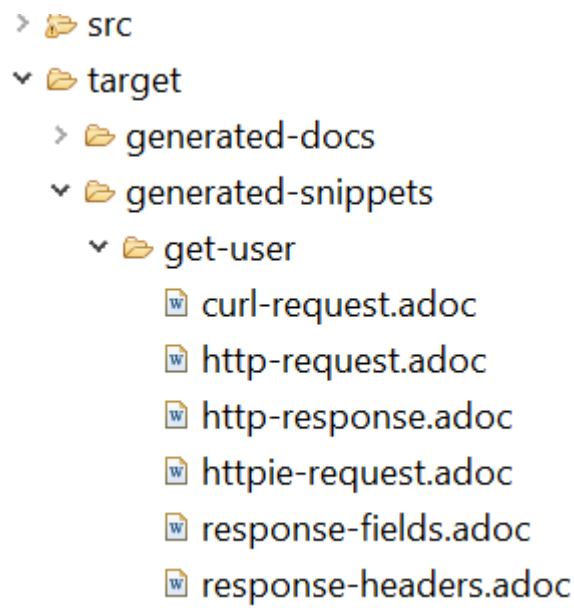
}

Давайте напишем тестовый пример, чтобы протестировать эту службу:

```
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
                        .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.eyJzdWIiOiJiaGFyZHdhai5uaXRpc2gxOSIsInJvbGVzIjpbIkFETUlOIl0sImlzcyI6Imh0dHA6Ly9ob3

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

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

                        responseFields(
                                fieldWithPath("username").description("Unique name for the
record"), //Asserts that the response has 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()
                        )
                        ));
    }
}
```

Запустите единичный тест, и некоторые файлы будут созданы в целевой папке.



Создайте исходную папку как **src / main / asciidocs** и создайте **файл** doc **с префиксом adoc** для документирования ваших данных о сервисе. Пример файла doc

```
[[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']
```

Тег include в файле doc должен включать в себя фрагменты. Вы можете указать любой формат, который вам нравится, для создания документа.

Когда у вас будет все на месте, **запустите Maven Build** . Он выполнит ваши тесты, и плагин asciidoc сгенерирует ваш html-файл **документа в** папке **target.generate-docs** . Ваш сгенерированный файл будет выглядеть примерно так:

Table of Contents

Retrieve User

A `GET` request gets a

Response fields

Path
username
password
securityAnswer
securityQuestion
email
roles
id

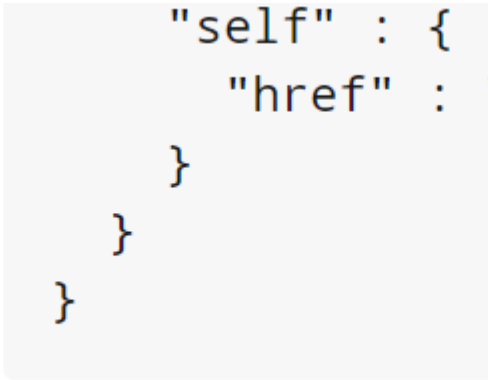
Table of Contents

Example request

```
$ curl 'http://r
'Authorization:
eyJhbGciOiJIUzUx
h0dHA6Ly9ob3NwaX
lHdJjaaodsIzKX4y
```

Example respons

```
HTTP/1.1 200 OK
Content-Type: ap
Content-Length:

{
  "password" : "
  "securityQuest
  "securityAnswe
  "id" : "591310
  "username" : "
  "email" : "bha
  "roles" : [ "A
  "_links" : {
    "self" : {
```



Всякий раз, когда выполняется ваша сборка maven, будет создан ваш последний документ, который всегда будет соответствовать вашим услугам. Просто опубликуйте этот документ для потребителей / клиентов службы.

Счастливое кодирование.

Прочитайте Начало работы с микросервисами онлайн:
https://riptutorial.com/ru/microservices/topic/7719/начало-работы-с-микросервисами

глава 2: API-шлюз

Вступление

Архитектура микросервисов обеспечивает большую гибкость для развязки приложений и разработки независимых приложений. Микросервис всегда должен быть независимо тестируемым и развертываемым.

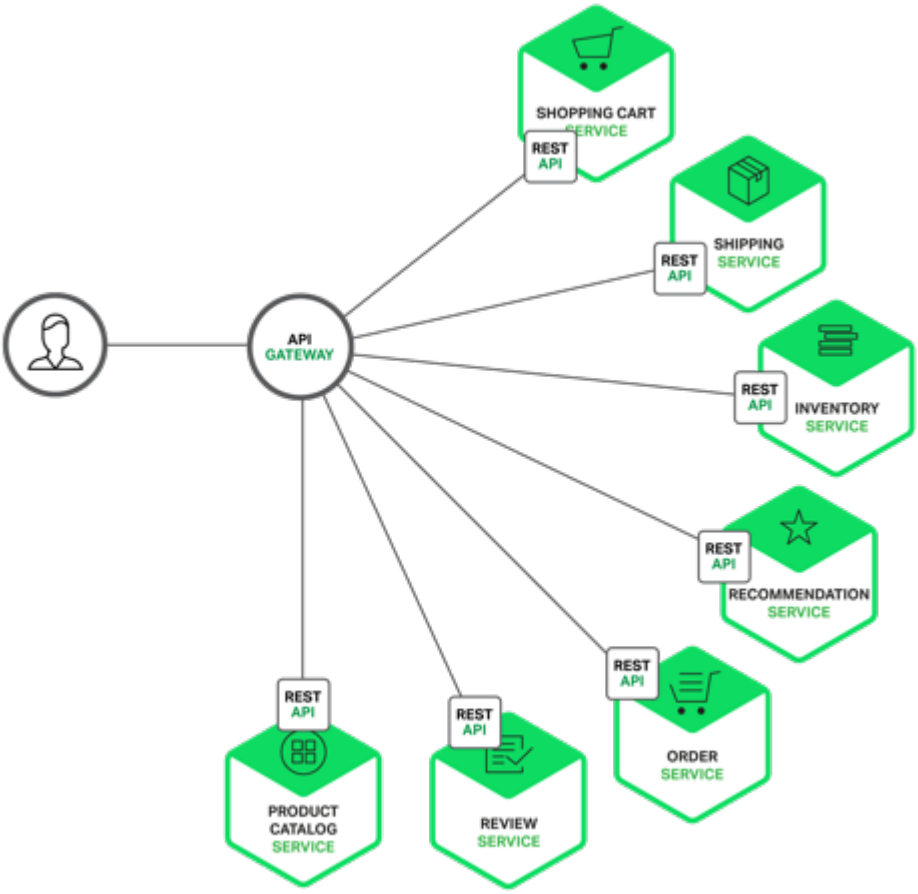
Но, поскольку у вас слишком много сервисов, **необходимо иметь API-шлюз** .

Вы не можете предоставлять все свои услуги внешним клиентам. Вы должны иметь некоторый слой абстракции, который действует как привратник для всех ваших Microservices. Одна точка входа для всех ваших услуг.

Examples

обзор

Предположим, у вас есть облако электронной коммерции, имеющее различные службы Microservices в качестве службы корзины покупок, услугу заказа, инвентаризации и т. Д. У вас должен быть API-шлюз в качестве службы Edge для внешнего мира.



Шлюз API аннотирует детали (хост и порт) о подстроечных Microservices. Теперь все ваши клиенты просто должны знать один URL-адрес сервера, который является вашим шлюзом API. Любые изменения в любом другом Miroservice не потребуют каких-либо изменений в вашем клиентском приложении. Всякий раз, когда шлюз API получает запрос, **он направляет запрос на конкретный Microservice** .

Прочитайте API-шлюз онлайн: https://riptutorial.com/ru/microservices/topic/10904/api-шлюз

кредиты

S. No	Главы	Contributors
1	Начало работы с микросервисами	Community, Dinusha, mohan08p, Nitish Bhardwaj
2	API-шлюз	Nitish Bhardwaj