배우기

microservices

#microservices

1:

,   .

.   .

Examples

Microservices

- CI / CD
-
- API
- API
-
-
-

API

Spring REST Docs .   .   .

.

Maven   .

pom .

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

build.plugins asciidoc .

```
<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>
```

.

```
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);
 }
```

```
        }
```

.

```
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;
```

```java
    @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.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 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()
                        )
                        ));
    }

}
```

. http://docs.spring.io/spring-restdocs/docs/current/reference/html5/

API

Spring REST Docs . . .

.

maven .

pom .

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

ASCII. docs 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>
```

.

```
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);
}

}
```

}

.

```
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.eyJzdWIiOiJiaGFyZHdhai5uaXRpc2gxOSIsInJvbGVzIjpbIkFETUlOIl0sImlzcyI6Imh0dHA6Ly9ob3N3
                .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()
                    )
                    ));
    }
}
```

> 📁 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

.

src / main / asciidocs   **adoc** doc   . 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']
```

doc **include** . .

Maven Build . asciidoc **target.generate-docs** html . .

Table of Contents

Retrieve User

A `GET` request gets a

Response fields

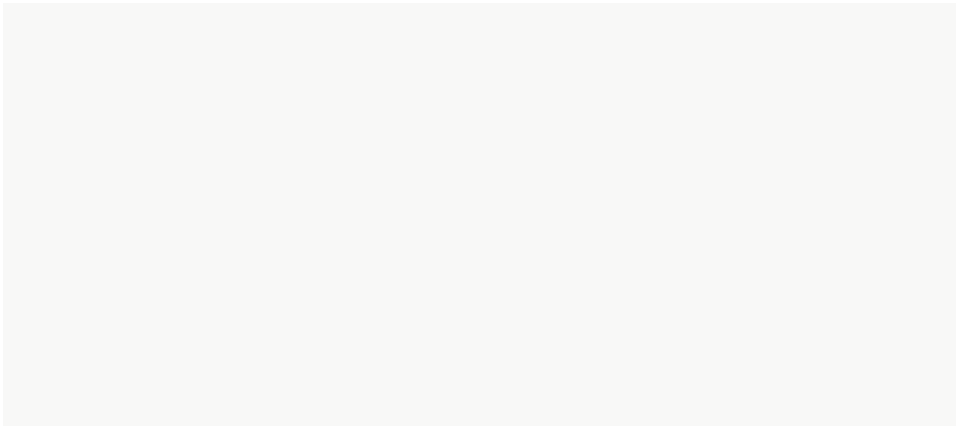| Path |
|------|
| username |
| password |
| securityAnswer |
| securityQuestion |
| email |
| roles |
| id |

Table of Contents

Example request

```
$ curl 'http://r
'Authorization:
eyJhbGciOiJIUzUx
hOdHA6Ly9ob3NwaX
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" : {
```

```
        "self" : {
          "href" :
        }
      }
    }
```
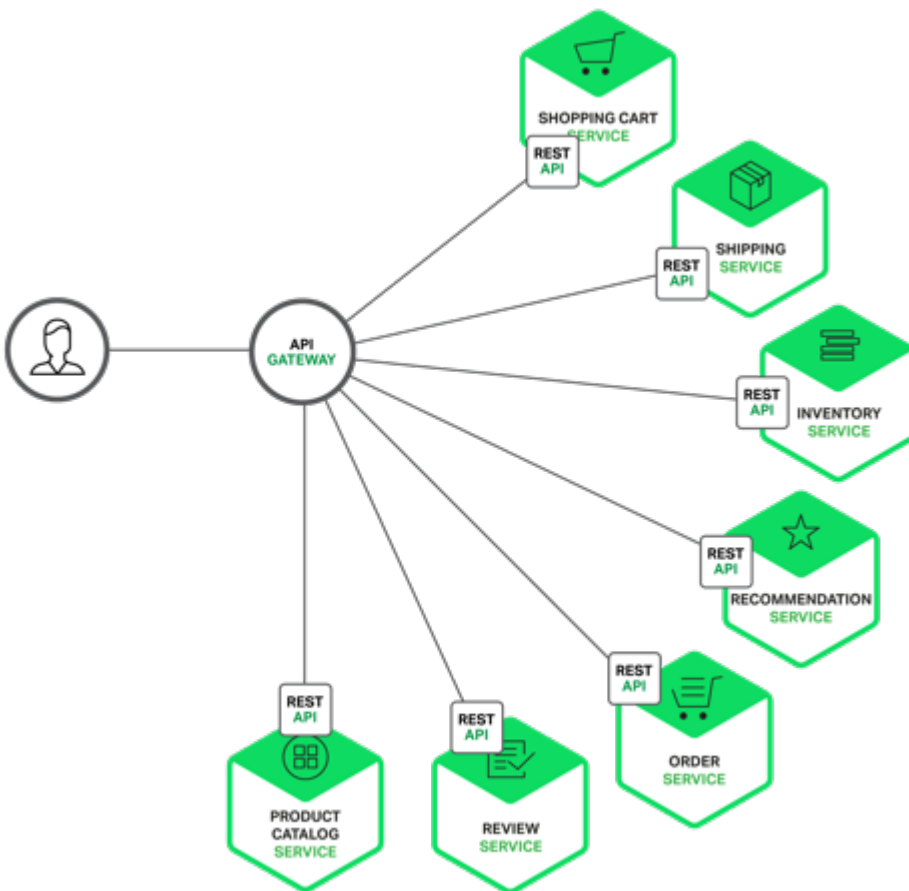
Maven   .   / .

.

2: API

Microservices   .   .

API   .

.   .   .

Examples

, ,   Microservices   .   API .

API   Microservices   ( ) .   API   URL . Miroservice   . API   **Microservice**   .

API   : https://riptutorial.com/ko/microservices/topic/10904/api-

S. No		Contributors
1		Community, Dinusha, mohan08p, Nitish Bhardwaj
2	API	Nitish Bhardwaj