记录时间：2019.09.23
公司名称： 火鹰科技有限公司
学生姓名： 曾伟涛
记录类型： 学习笔记
记录阶段如下：

2019.09.23-2019.09.30

springBoot项目下利用Swagger实现api文档

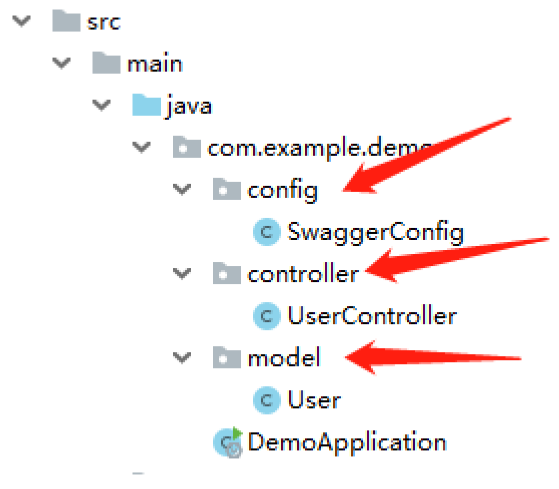
以Intelli IDEA为例：

步骤1：创建springBoot项目（点击此链接跟着操作，不会回家睡觉去吧）
https://baijiahao.baidu.com/s?id=1632687280256687825&wfr=spider&for=pc

步骤2：在pom.xml内配置其依赖 --->目前最新是swagger 2

```xml
<dependency>
 <groupId>io.springfox</groupId>
 <artifactId>springfox-swagger2</artifactId>
 <version>2.7.0</version>
</dependency>
<dependency>
 <groupId>io.springfox</groupId>
 <artifactId>springfox-swagger-ui</artifactId>
 <version>2.7.0</version>
 </dependency>
```

步骤3：创建包及相应的类



以左图为例：在主包下分别创建 config controller model 这三个子包
在这三包下分别创建相应的类：
SwaggerConfig UserController User

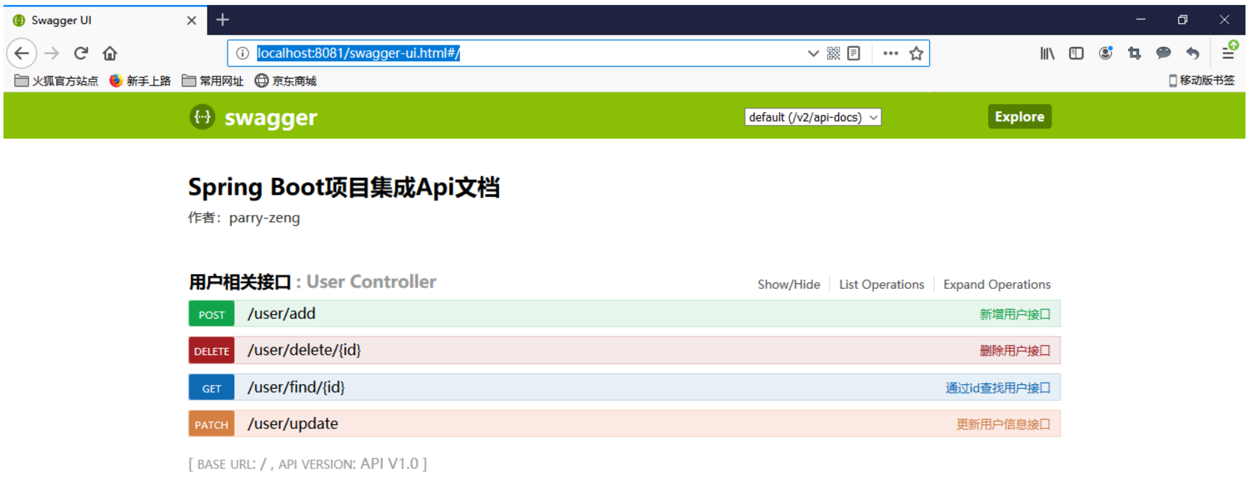
解析相应类的作用:
config包：SwaggerConfig类为swagger的配置类，即对swagger进项相应的配置
Controller包：UserController类为User的控制层，即对User的而数据进行增删改查等操作的处理

User包：User类为User数据实体类，如用户名，密码，地址，手机号码等

步骤4：编写相应类的内容，以UserController.java为例，其他请到https://github.com/jious/springBoot.git

```
@RestController
@RequestMapping("/user")
public class UserController {
    @PostMapping("/add") //PostMapping:请求数据类型的控制
    public boolean addUser(@RequestBody User user) { //@RequestBody 对请求的参数数据写在body体内，对数据进行安全保护
        return false;
    }
    @GetMapping("/find/{id}")
    public User findById(@PathVariable("id") int id) { //@PathVariable 绑定URL中参数到处理器方法形参中
         return new User();
     }
     @PutMapping("/update") or @PatchMaping("/update") //两者作用一样 请求类型 ： “更新”
     public boolean update(@RequestBody User user) {
         return true;
     }
     @DeleteMapping("/delete/{id}")
     public boolean delete(@PathVariable("id") int id) {
         return true;
     }
}
```

步骤5：运行idea，在浏览器输入 http://localhost:8081/swagger-ui.html/，会看到此页面（注意你自己写的port端口号：默认是8080）



Spring Boot项目集成Api文档

作者: parry-zeng

用户相关接口 : User Controller   Show/Hide | List Operations | Expand Operations

POST /user/add 新增用户接口

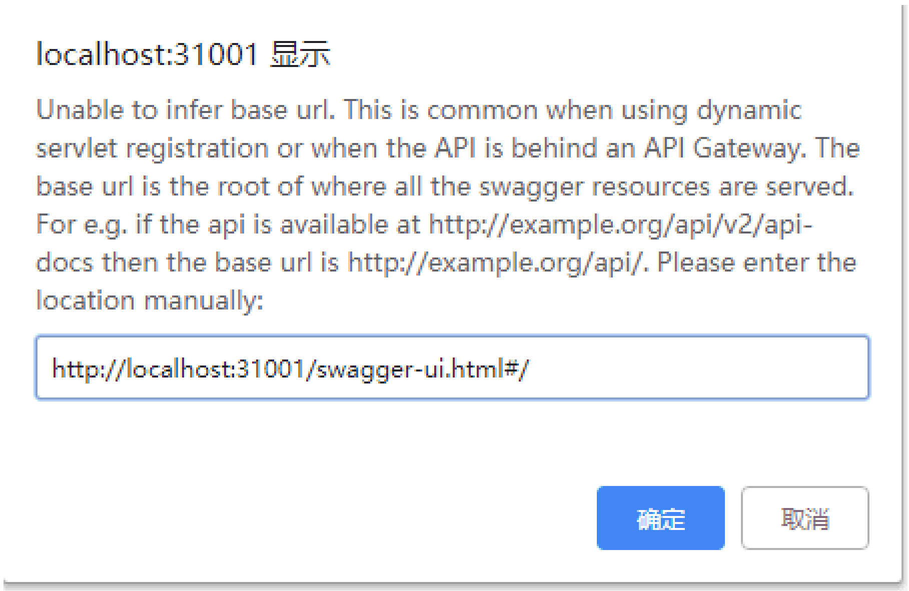
DELETE /user/delete/{id} 删除用户接口

GET /user/find/{id} 通过id查找用户接口

PATCH /user/update 更新用户信息接口

[BASE URL: / , API VERSION: API V1.0]

可能遇到的问题:



localhost:31001 显示

Unable to infer base url. This is common when using dynamic servlet registration or when the API is behind an API Gateway. The base url is the root of where all the swagger resources are served. For e.g. if the api is available at http://example.org/api/v2/api-docs then the base url is http://example.org/api/. Please enter the location manually:

http://localhost:31001/swagger-ui.html#/

出现这种情况，有以下几种原因:

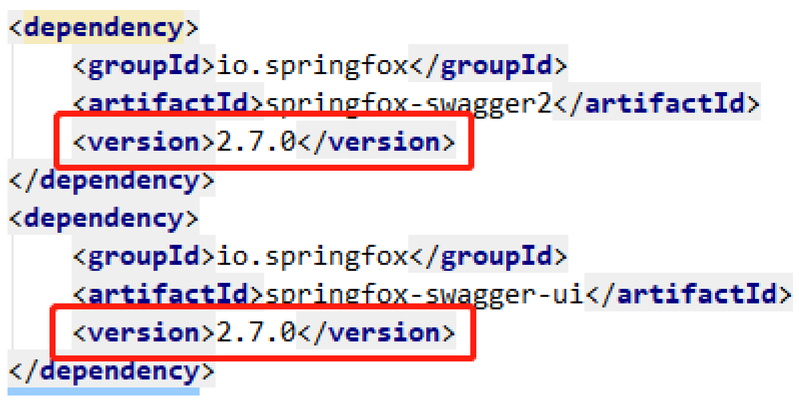
(1) 浏览器缓存问题

解决方案：清理相应浏览器的缓存或者换个浏览器

(2) swagger配置类代码错误:

比如注解的是@Configuration，你选择了@Configurable

(3) 配置swagger以及swagger-ui的依赖版本不一致



```xml
<dependency>
    <groupId>io.springfox</groupId>
    <artifactId>springfox-swagger2</artifactId>
    <version>2.7.0</version>
</dependency>
<dependency>
    <groupId>io.springfox</groupId>
    <artifactId>springfox-swagger-ui</artifactId>
    <version>2.7.0</version>
</dependency>
```

springboot整合mybatis实现连接数据库的增删改查

（话不多说，直接上代码）

在这上代码之前，需要有以下的准备：

1.电脑要有navicat for mysql这个软件

2.在idea的settings -->plugs里安装**lombok**这个插件

这两个准备完成了，开始看代码：

上面的**UserController**有所改动：

```java
import boat.commons.orika.Orika;
import boat.web.response.CloudApiResponse;
import io.swagger.annotations.Api;
import io.swagger.annotations.ApiOperation;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.MediaType;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.annotation.*;
import test.api.UserApiPost;
 import test.model.UserModel;
 import test.api.UserApiPatch;
 import test.projection.UserProjection;
 import test.service.UserService;

 import static jdk.nashorn.internal.objects.NativeDebug.map;

 @Api(tags = "用户列表-API")// 标题
  @RequestMapping("user:user")//请求路径
  @RestController
  public class userController {
  @Autowired
  private UserService userService;

```

```java
@ApiOperation(httpMethod = "POST", value = "增加", produces = MediaType.APPLICATION_JSON_UTF8_VALUE, consumes = MediaType.APPLICATION_JSON_UTF8_VALUE)
@PostMapping(value = "", produces = MediaType.APPLICATION_JSON_UTF8_VALUE, consumes = MediaType.APPLICATION_JSON_UTF8_VALUE)
CloudApiResponse<UserProjection> create(@Validated @RequestBody UserApiPost api) {
CloudApiResponse<UserProjection> apiResponse = new CloudApiResponse<>();
UserModel resultModel = userService.save(Orika.map(api, UserModel.class));
apiResponse.setData(resultModel, UserProjection.class);
apiResponse.setMsg("添加成功");
return apiResponse;
}

@ApiOperation(httpMethod = "PATCH", value = "更新", produces = MediaType.APPLICATION_JSON_UTF8_VALUE, consumes = MediaType.APPLICATION_JSON_UTF8_VALUE)
@PatchMapping(value = "/{id}", produces = MediaType.APPLICATION_JSON_UTF8_VALUE, consumes = MediaType.APPLICATION_JSON_UTF8_VALUE)
CloudApiResponse<UserProjection> update(@PathVariable Integer id,
@Validated @RequestBody UserApiPatch api) {
CloudApiResponse<UserProjection> apiResponse = new CloudApiResponse<>();
UserModel userModel = Orika.map(api, UserModel.class);
userModel.setId(id);
UserModel resultModel = userService.update(userModel);
apiResponse.setData(resultModel, UserProjection.class);
apiResponse.setMsg("更新成功");
return apiResponse;
}

@ApiOperation(httpMethod = "DELETE", value = "删除", produces = MediaType.APPLICATION_JSON_UTF8_VALUE)
@DeleteMapping(value = "/{id}", produces = MediaType.APPLICATION_JSON_UTF8_VALUE)
public CloudApiResponse<UserProjection> delete(@PathVariable Integer id) {
CloudApiResponse<UserProjection> apiResponse = new CloudApiResponse<>();
UserModel userModel = new UserModel();
userModel.setId(id);
userService.deleteById(userModel);
apiResponse.setMsg("删除成功");
return apiResponse;
}

@ApiOperation(httpMethod = "GET", value = "查询", produces = MediaType.APPLICATION_JSON_UTF8_VALUE)
@GetMapping(value = "/{id}", produces = MediaType.APPLICATION_JSON_UTF8_VALUE)
CloudApiResponse<UserProjection> findById(@PathVariable Integer id) {
CloudApiResponse<UserProjection> apiResponse = new CloudApiResponse<>();
UserModel model = userService.findById(id);
apiResponse.setData(model, UserProjection.class);
return apiResponse;
}
}
```

pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

 <modelVersion>4.0.0</modelVersion>
 <artifactId>gj-service</artifactId>
 <version>0.0.1-SNAPSHOT</version>

 <name></name>
 <description>Demo project for Spring Boot</description>


 <parent>
 <groupId>cn.figo.cloud</groupId>
 <artifactId>cloud-parent</artifactId>
 <version>Dalston.SR4</version>
 <relativePath/> <!-- lookup parent from repository -->
 </parent>


 <dependencies>
 <dependency>
 <groupId>org.springframework.boot</groupId>
```

```xml
<artifactId>spring-boot-starter-test</artifactId>
<scope>test</scope>
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-devtools</artifactId>
<scope>runtime</scope>
</dependency>
<!-- security-code -->
<dependency>
<groupId>org.springframework.security</groupId>
<artifactId>spring-security-core</artifactId>
<optional>true</optional>
</dependency>
<!-- kafka -->
<dependency>
<groupId>org.springframework.kafka</groupId>
<artifactId>spring-kafka</artifactId>
</dependency>
<!-- redis -->
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-data-redis</artifactId>
</dependency>
<!-- boat-mybatis -->
<dependency>
<groupId>cn.figo.cloud</groupId>
<artifactId>boat-mybatis</artifactId>
<version>0.0.1</version>
</dependency>
<!-- boat-web -->
<dependency>
<groupId>cn.figo.cloud</groupId>
<artifactId>boat-web</artifactId>
<version>0.0.1</version>
<optional>true</optional>
```

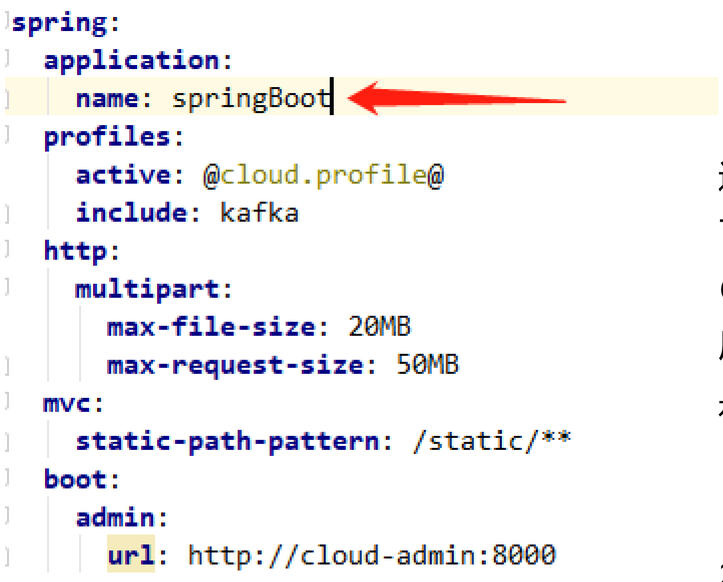
```xml
 </dependency>
 <!-- vendor-third-oauth -->
 <dependency>
 <groupId>cn.figo.cloud</groupId>
 <artifactId>vendor-third-oauth</artifactId>
 <version>0.0.1</version>
 </dependency>
 <!-- vendor-jpush -->
 <dependency>
 <groupId>cn.figo.cloud</groupId>
 <artifactId>vendor-jpush</artifactId>
 <version>0.0.1</version>
 </dependency>

 </dependencies>
</project>
```

在复制这个maven库包要注意：**cn.figo**这样的字样均为公司的包，我调用的是公司包

其他代码请上这个链接：https://github.com/jious/springBoot.git中的springBoot文件

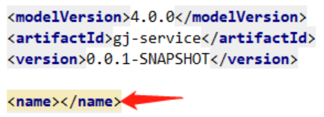
注意事项：

1.application.yml:



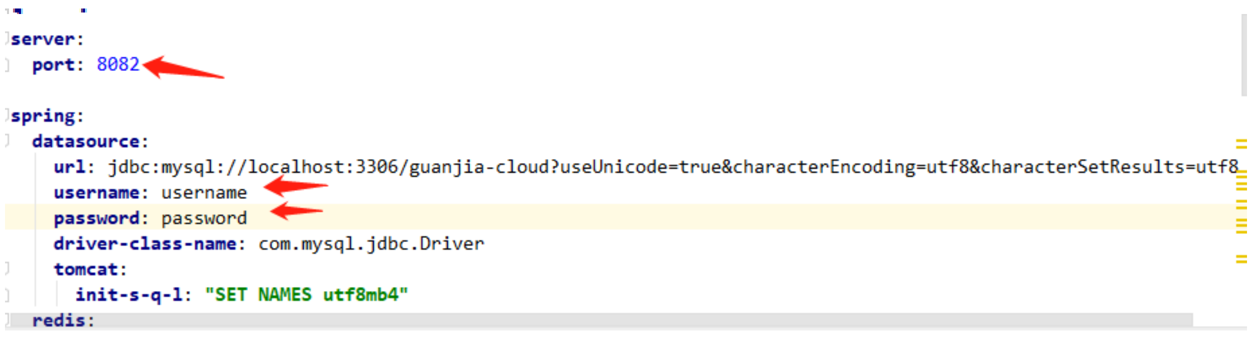
```yaml
spring:
  application:
    name: springBoot
  profiles:
    active: @cloud.profile@
    include: kafka
  http:
    multipart:
      max-file-size: 20MB
      max-request-size: 50MB
  mvc:
    static-path-pattern: /static/**
  boot:
    admin:
      url: http://cloud-admin:8000
```

这里的name值可以自定义，它的值决定了你启动的路径，也可以@project.name@这样的格式，这将调用你pom.xml文件里的name值，我这里是为空



```xml
<modelVersion>4.0.0</modelVersion>
<artifactId>gj-service</artifactId>
<version>0.0.1-SNAPSHOT</version>

<name></name>
```

2.application-dev.yml:

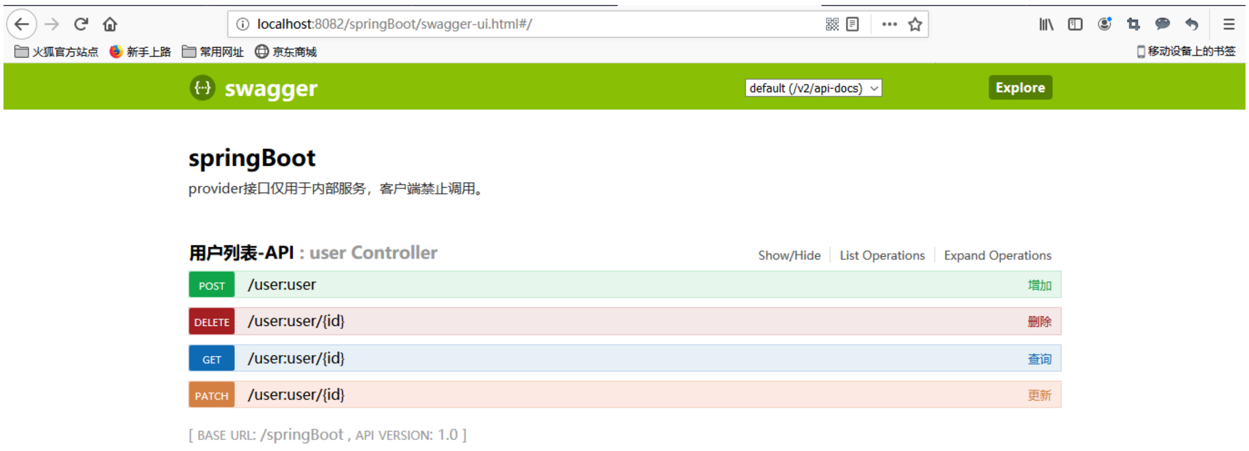


```yaml
server:
  port: 8082

spring:
  datasource:
    url: jdbc:mysql://localhost:3306/guanjia-cloud?useUnicode=true&characterEncoding=utf8&characterSetResults=utf8
    username: username
    password: password
    driver-class-name: com.mysql.jdbc.Driver
    tomcat:
      init-s-q-l: "SET NAMES utf8mb4"
  redis:
```

我用的端口是**8082**，注意你们写的**数据库名以及表名，用户名 密码**等

代码写完，校对与我是否一致，一致后请看一下效果是否能实现：

在浏览器里输入：localhost:8082/springBoot/swagger-ui.html



localhost:8082/springBoot/swagger-ui.html#/

火狐官方站点 新手上路 常用网址 京东商城 移动设备上的书签

swagger default (/v2/api-docs) Explore

springBoot

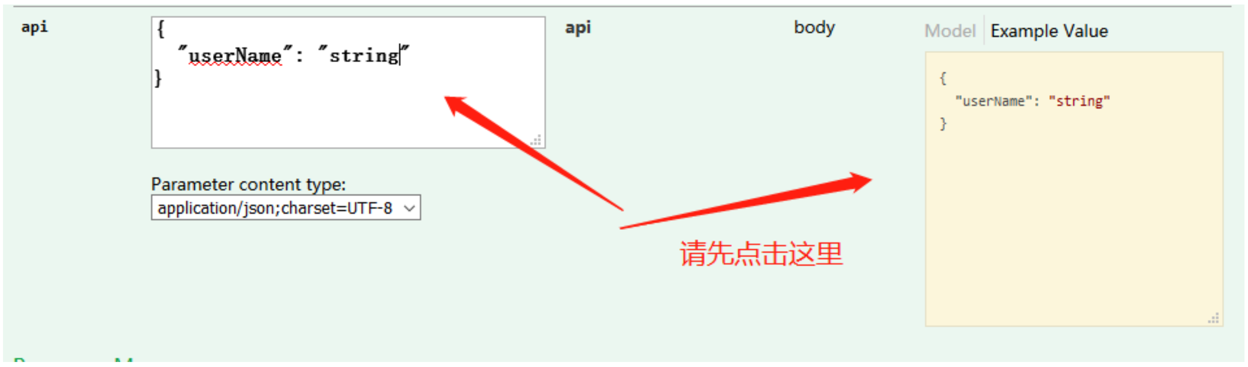
provider接口仅用于内部服务，客户端禁止调用。

用户列表-API : user Controller Show/Hide | List Operations | Expand Operations

POST	/user:user	增加
DELETE	/user:user/{id}	删除
GET	/user:user/{id}	查询
PATCH	/user:user/{id}	更新

[BASE URL: /springBoot , API VERSION: 1.0]

增加：



api

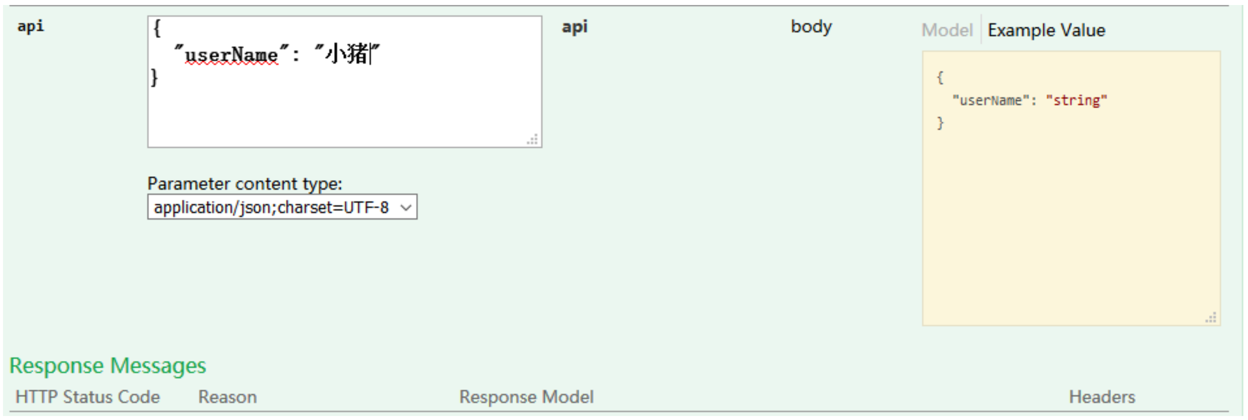
```
{
  "userName": "string"
}
```

api body Model Example Value

```
{
  "userName": "string"
}
```

Parameter content type: application/json;charset=UTF-8

我们在那里输入 “小猪” ，然后点击try it out



api

```
{
  "userName": "小猪"
}
```

api body Model Example Value

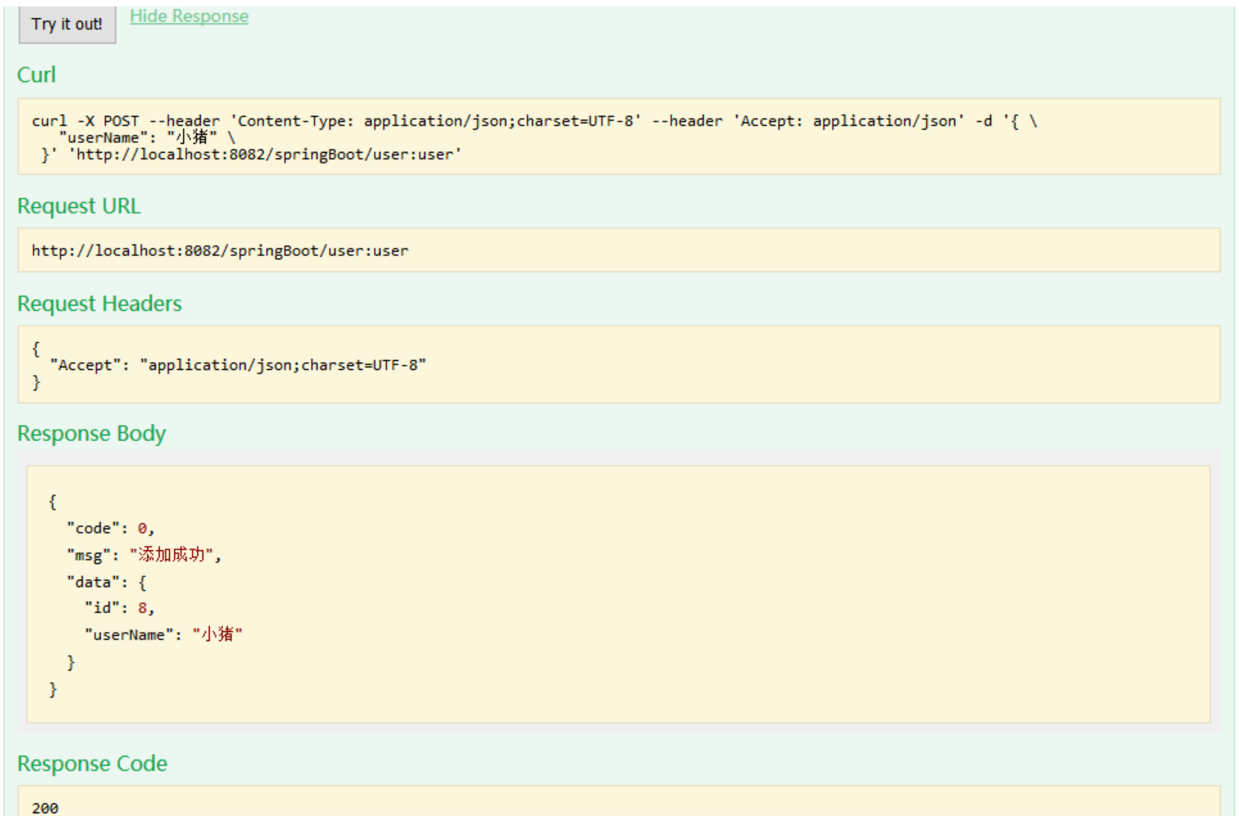
```
{
  "userName": "string"
}
```

Parameter content type: application/json;charset=UTF-8

Response Messages

HTTP Status Code Reason Response Model Headers

结果如下：



Curl

```
curl -X POST --header 'Content-Type: application/json;charset=UTF-8' --header 'Accept: application/json' -d '{ \
   "userName": "小猪" \
 }' 'http://localhost:8082/springBoot/user:user'
```

Request URL

```
http://localhost:8082/springBoot/user:user
```

Request Headers

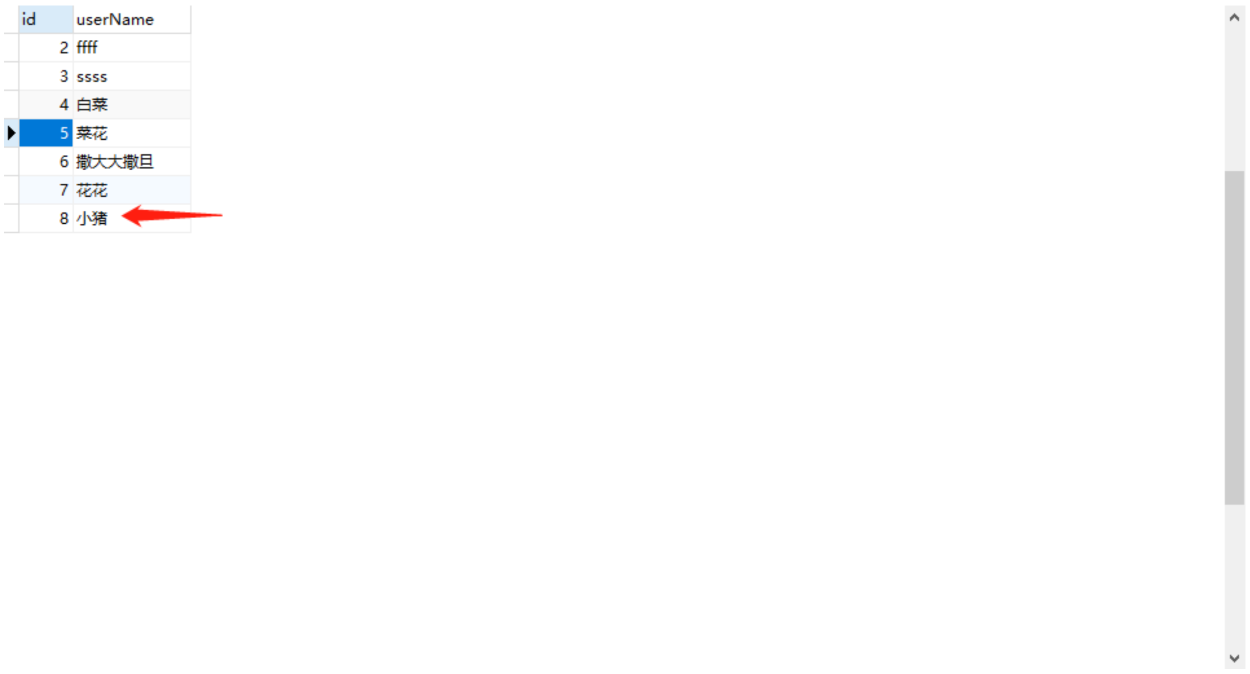
```
{
  "Accept": "application/json;charset=UTF-8"
}
```

Response Body

```
{
  "code": 0,
  "msg": "添加成功",
  "data": {
    "id": 8,
    "userName": "小猪"
  }
}
```

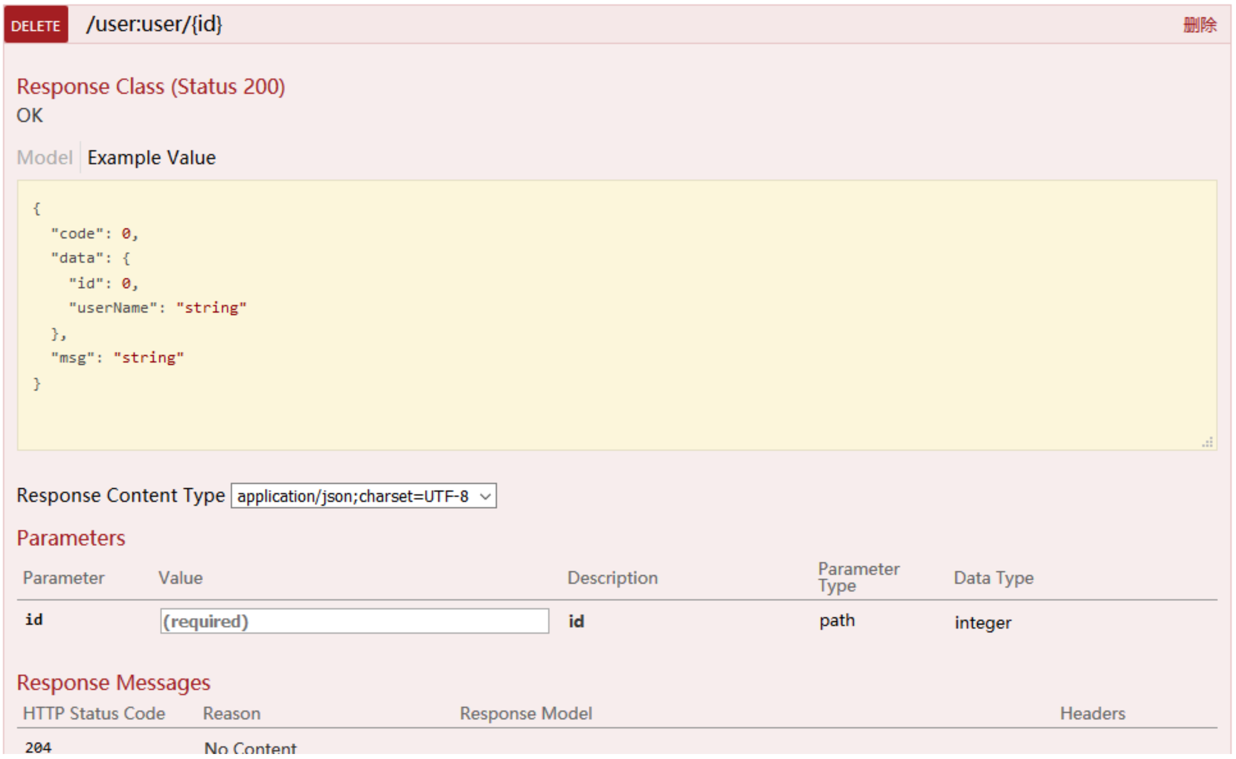
Response Code

```
200
```



id	userName
2	ffff
3	ssss
4	白菜
5	菜花
6	撒大大撒旦
7	花花
8	小猪

删除：



/user:user/{id}

Response Class (Status 200)
OK

```
{
  "code": 0,
  "data": {
    "id": 0,
    "userName": "string"
  },
  "msg": "string"
}
```

Response Content Type application/json;charset=UTF-8

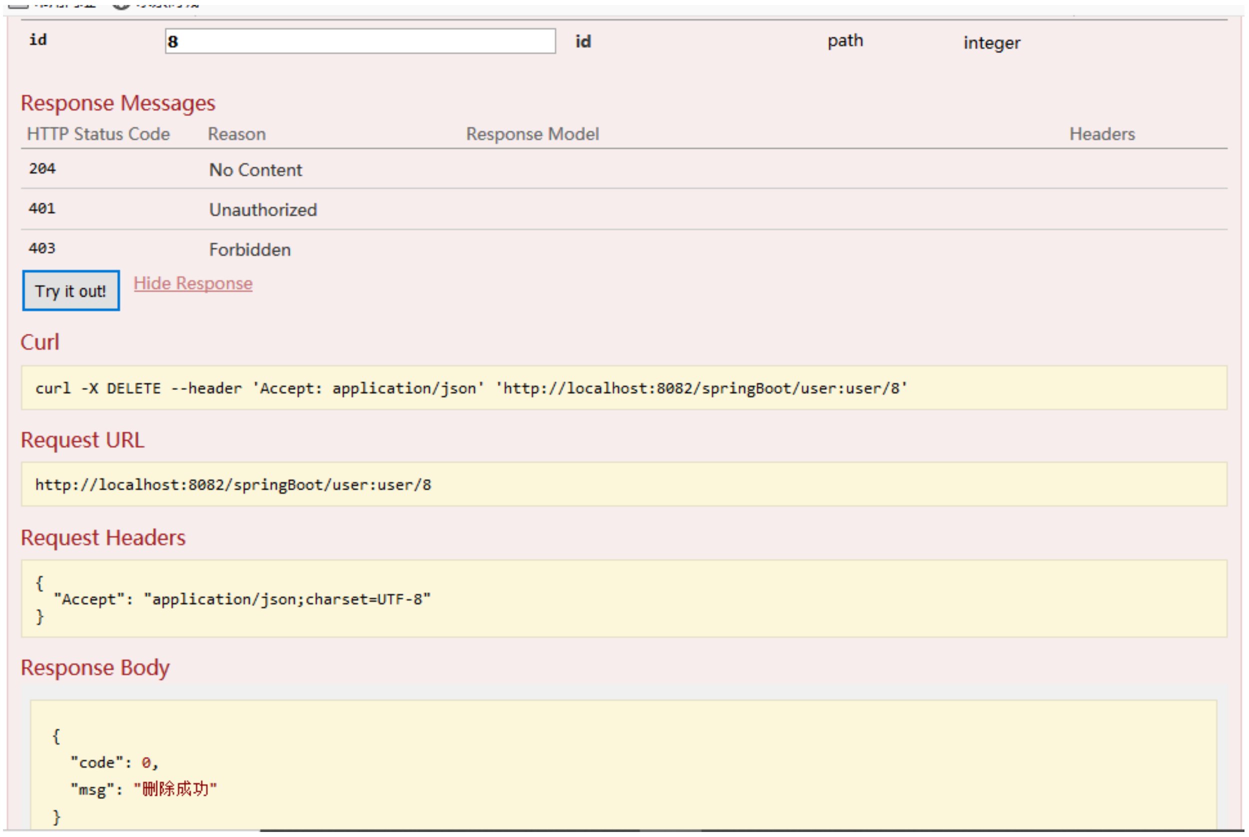
Parameters

Parameter	Value	Description	Parameter Type	Data Type
id	(required)	id	path	integer

Response Messages

HTTP Status Code	Reason	Response Model	Headers
204	No Content		

在上面添加可以看到“小猪”的id是8，我们来测试删除“小猪”，在上面的id里输入8



Parameter	Value	Description	Parameter Type	Data Type
id	8	id	path	integer

Response Messages

HTTP Status Code	Reason	Response Model	Headers
204	No Content		
401	Unauthorized		
403	Forbidden		

Curl

```
curl -X DELETE --header 'Accept: application/json' 'http://localhost:8082/springBoot/user:user/8'
```

Request URL

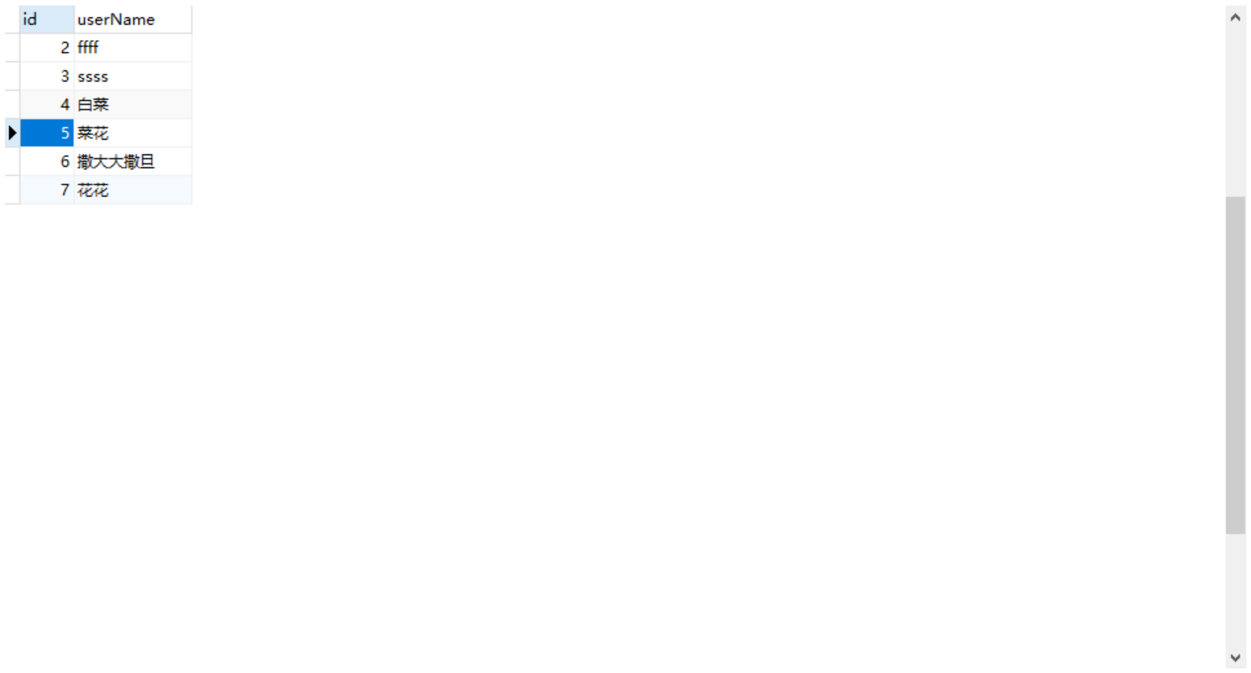
```
http://localhost:8082/springBoot/user:user/8
```

Request Headers

```
{
  "Accept": "application/json;charset=UTF-8"
}
```

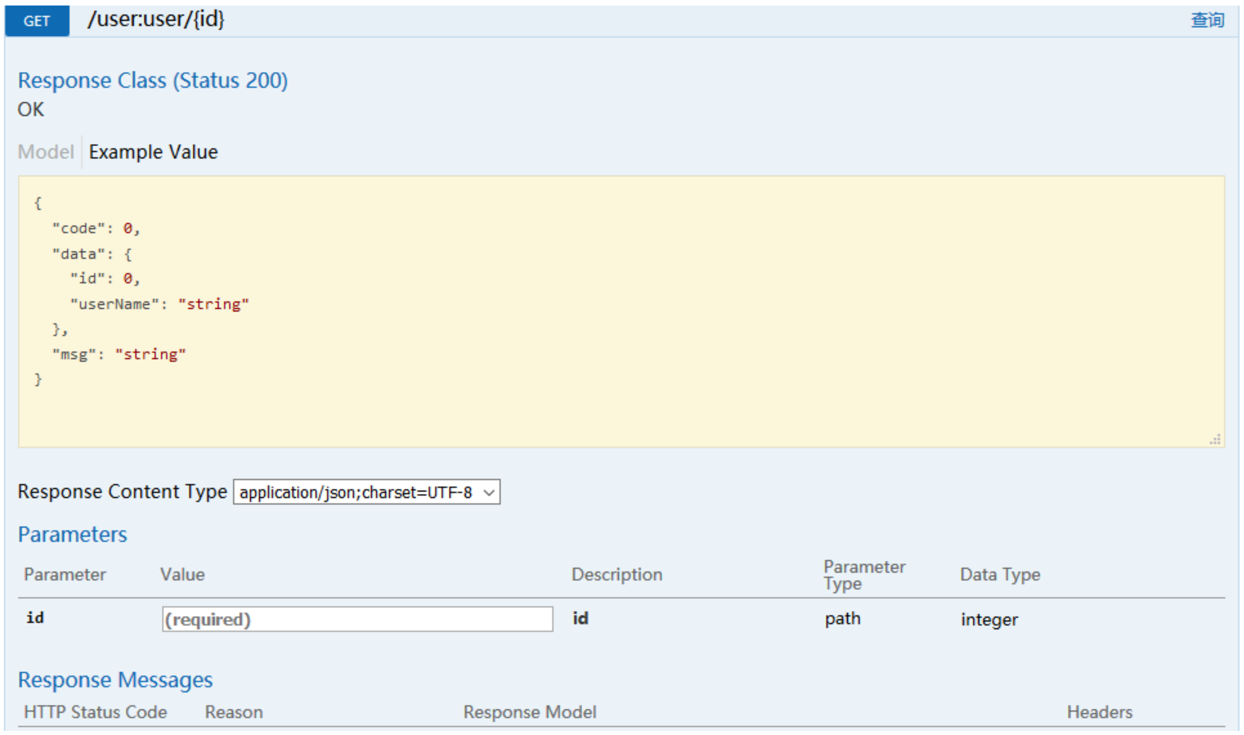
Response Body

```
{
  "code": 0,
  "msg": "删除成功"
}
```



id	userName
2	ffff
3	ssss
4	白菜
5	菜花
6	撒大大撒旦
7	花花

查询：



GET /user:user/{id} 查询

Response Class (Status 200)
OK

Model Example Value

```
{
  "code": 0,
  "data": {
    "id": 0,
    "userName": "string"
  },
  "msg": "string"
}
```

Response Content Type application/json;charset=UTF-8

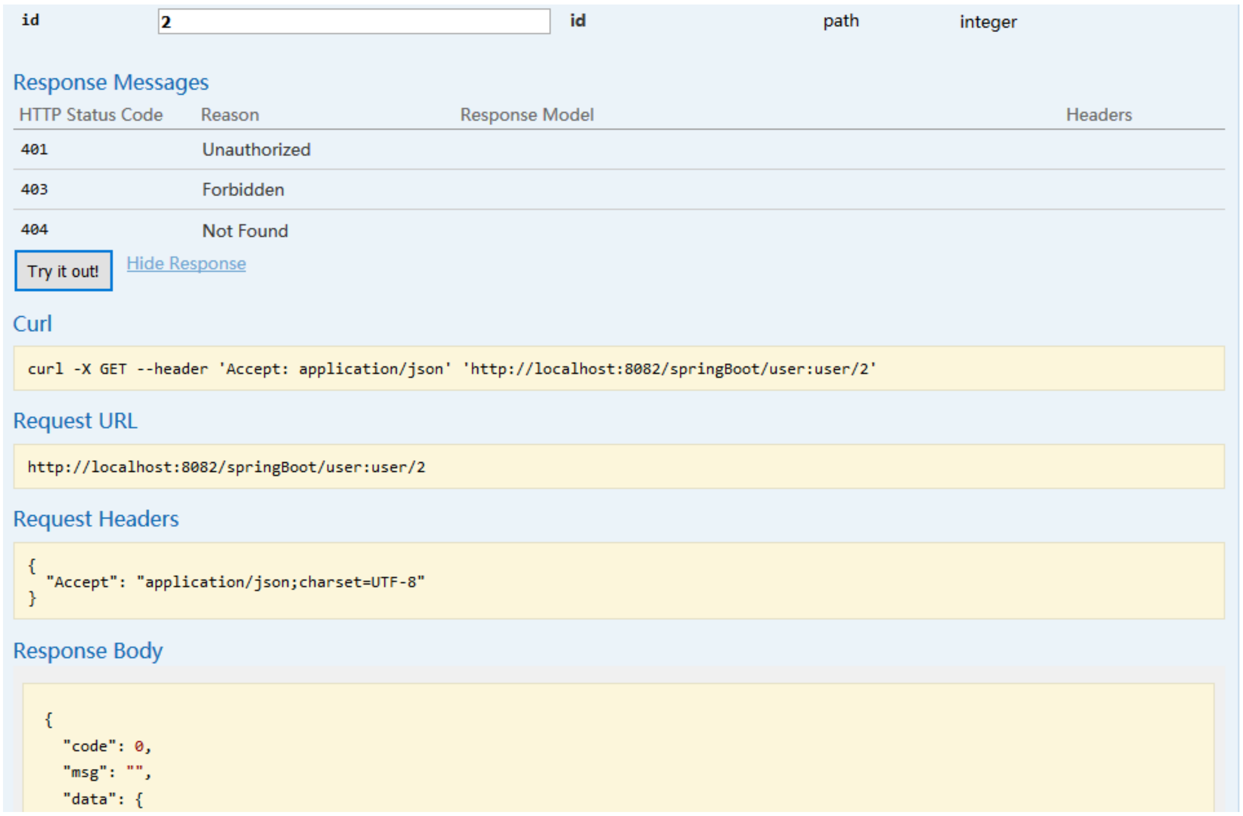
Parameters

Parameter	Value	Description	Parameter Type	Data Type
id	(required)	id	path	integer

Response Messages

HTTP Status Code	Reason	Response Model	Headers

我们来查询1，看查出数据是否与数据库一致：



id	2	id	path	integer

Response Messages

HTTP Status Code	Reason	Response Model	Headers
401	Unauthorized		
403	Forbidden		
404	Not Found		

Curl

```
curl -X GET --header 'Accept: application/json' 'http://localhost:8082/springBoot/user:user/2'
```

Request URL

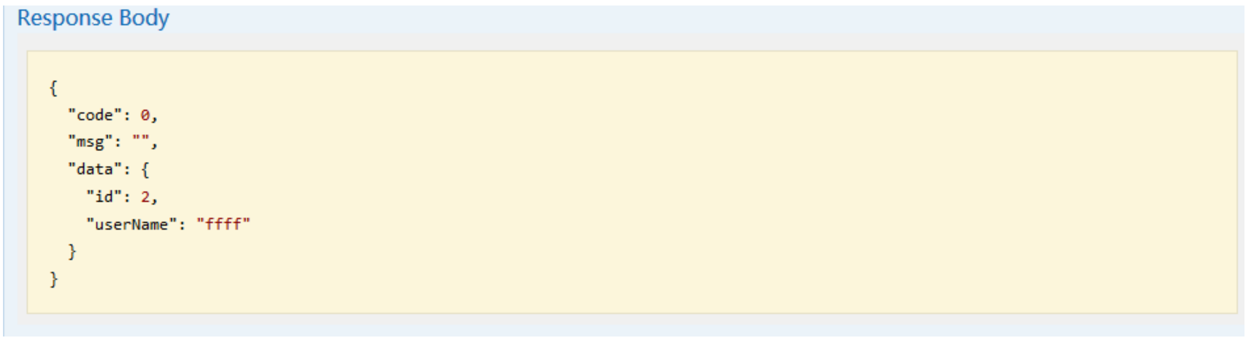
```
http://localhost:8082/springBoot/user:user/2
```

Request Headers

```
{
  "Accept": "application/json;charset=UTF-8"
}
```

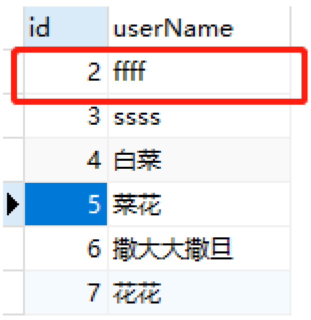
Response Body

```
{
  "code": 0,
  "msg": "",
  "data": {
```



Response Body

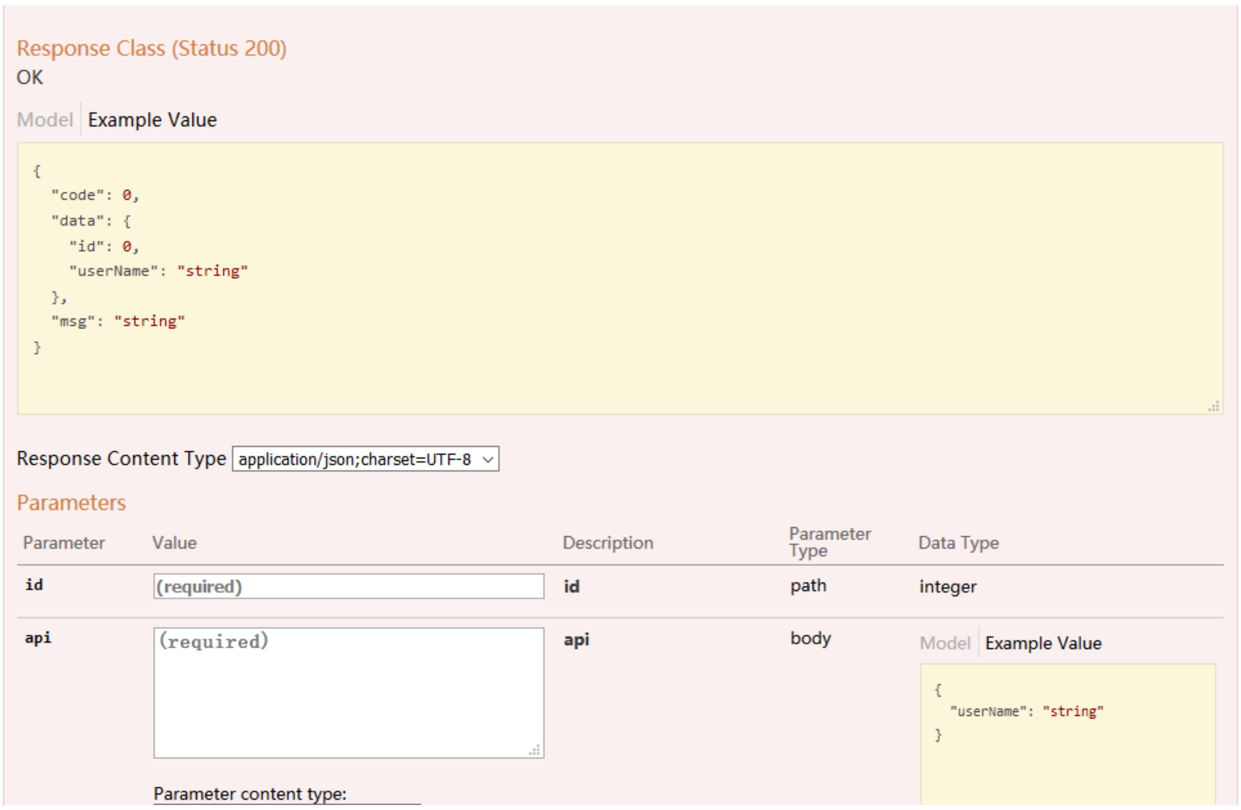
```
{
  "code": 0,
  "msg": "",
  "data": {
    "id": 2,
    "userName": "ffff"
  }
}
```



id	userName
2	ffff
3	ssss
4	白菜
5	菜花
6	撒大大撒旦
7	花花

与数据库一致，成功

更新（修改）：



Response Class (Status 200)

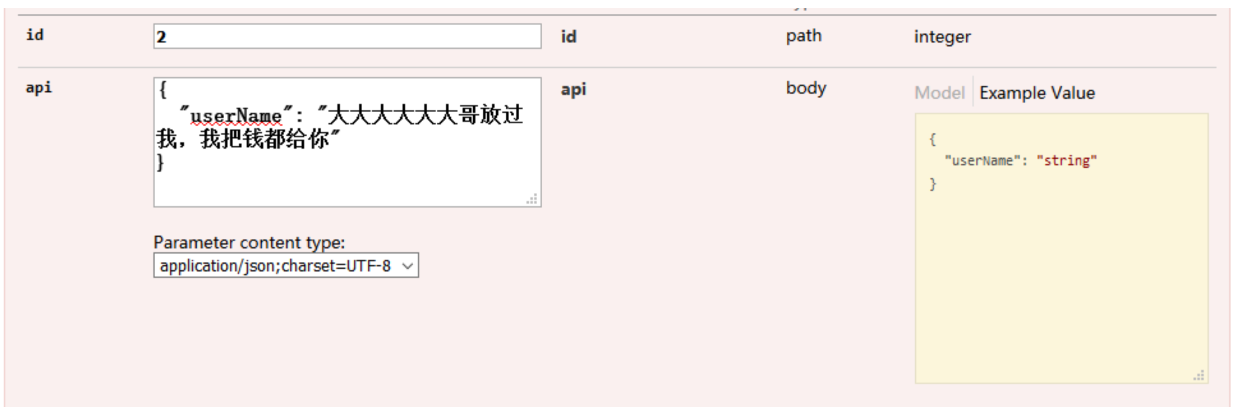
OK

```
{
  "code": 0,
  "data": {
    "id": 0,
    "userName": "string"
  },
  "msg": "string"
}
```

Response Content Type: application/json;charset=UTF-8

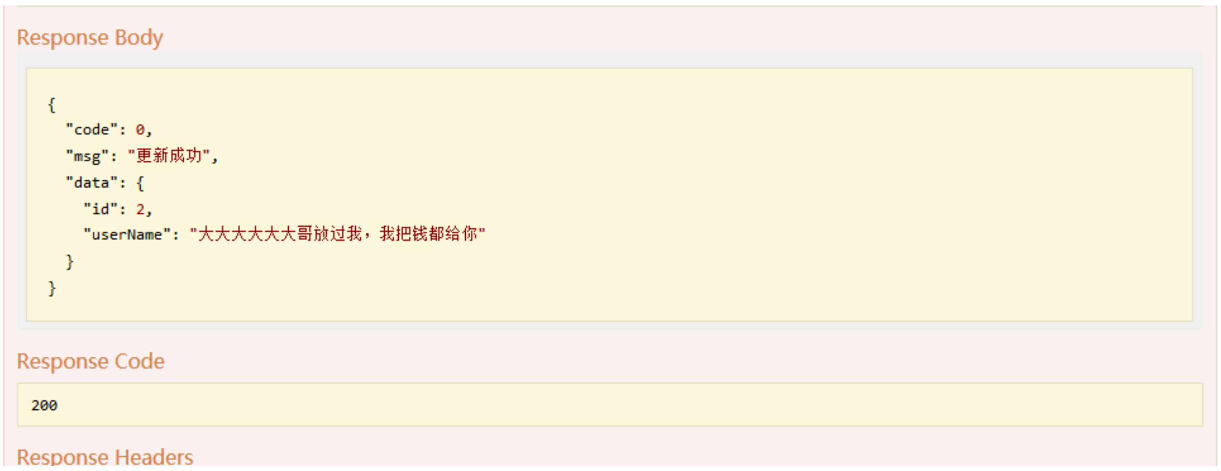
Parameters

Parameter	Value	Description	Parameter Type	Data Type
id	(required)	id	path	integer
api	(required) Parameter content type:	api	body	Model / Example Value `{ "userName": "string" }`



id	2	id	path	integer

api	{
	"userName": "大大大大大大哥放过我，我把钱都给你"
	}
	Parameter content type:
	application/json;charset=UTF-8
		api	body	Model / Example Value
				`{ "userName": "string" }`



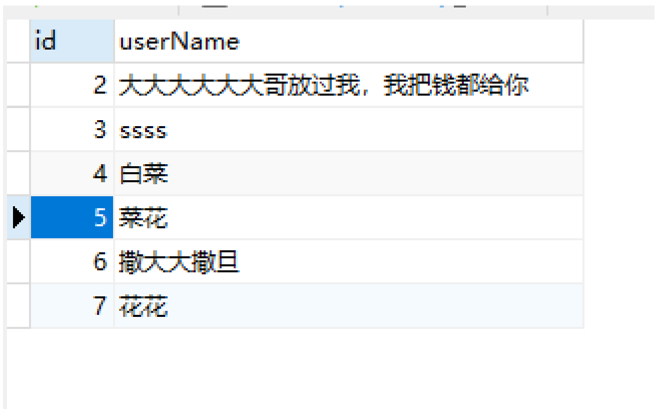
Response Body

```
{
  "code": 0,
  "msg": "更新成功",
  "data": {
    "id": 2,
    "userName": "大大大大大大哥放过我，我把钱都给你"
  }
}
```

Response Code

```
200
```

Response Headers



id	userName
2	大大大大大大哥放过我，我把钱都给你
3	ssss
4	白菜
5	菜花
6	撒大大撒旦
7	花花

快动起手来实现吧！！！