Swagger

一、Swagger简介

1. 后端时代

前端只用管理静态页面，然后将html页面交给后端。后端使用模板引擎jsp生成页面。后端是主力

2. 前后端分离时代

- 后端：包括控制层，服务层，数据访问层 【后端团队负责】
- 前端：前端控制层，视图层 【前端团队负责】
    - 伪造后端数据：json。假数据已经存在，不需要后端
- 前后端通过API进行交互，restful
- 前后端相对独立，只通过json字符串交互，耦合度较低，
- 前后端甚至可以部署到不同的项目中

3.产生的问题

前后端集成联调，前后端人员无法做到很好的沟通

解决方法：

- 指定计划提纲，实时更新最新的API，降低集成的风险；
- 前后端分离：
    - 使用postman进行测试

二. Swagger

2.1、概念：

号称世界上最流行的API框架，RestFul API文档在线自动生成工具=》API文档与API定义同步更新。而且可以直接运行，可以在线测试API接口。支持多种语言。

2.2、官网

https://swagger.io/

2.3、依赖

- swagger2
- ui

2.4、SpringBoot项目集成swagger2

2.4.1、创建Springboot工程

POM

```
<?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>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.0.5.RELEASE</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.swagger</groupId>
    <artifactId>swagger-demo</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>swagger-demo</name>
    <description>Demo project for Spring Boot</description>

    <properties>
        <java.version>1.8</java.version>
    </properties>

    <dependencies>

        <!--swagger依赖-->
        <dependency>
            <groupId>io.springfox</groupId>
            <artifactId>springfox-swagger2</artifactId>
            <version>2.9.2</version>
        </dependency>

        <dependency>
            <groupId>io.springfox</groupId>
            <artifactId>springfox-swagger-ui</artifactId>
            <version>2.9.2</version>
        </dependency>

        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
```

```
            <version>1.16.20</version>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
            <exclusions>
                <exclusion>
                    <groupId>org.junit.vintage</groupId>
                    <artifactId>junit-vintage-engine</artifactId>
                </exclusion>
            </exclusions>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>

</project>
```

启动类

```
package com.swagger;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class SwaggerDemoApplication {

    public static void main(String[] args) {
        SpringApplication.run(SwaggerDemoApplication.class, args);
    }

}
```

测试类

```
package com.swagger.controller;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping("/hello")
public class SwaggerController {

    @GetMapping("/world")
    public String hello(){
        return "你好，世界";
    }
}
```

2.4.2、 Swagger2的相关依赖

```
<dependency>
      <groupId>io.springfox</groupId>
      <artifactId>springfox-swagger2</artifactId>
        <version>2.9.2</version>
</dependency>

<dependency>
      <groupId>io.springfox</groupId>
      <artifactId>springfox-swagger-ui</artifactId>
      <version>2.9.2</version>
 </dependency>
```

2.4.3、 编写配置类

Swagger2

```java
package com.swagger.config;

import org.springframework.context.annotation.Configuration;
import springfox.documentation.swagger2.annotations.EnableSwagger2;

@Configuration
@EnableSwagger2
public class SwaggerConfig {
}
```

2.4.4、测试

http://localhost:8080/swagger-ui.html

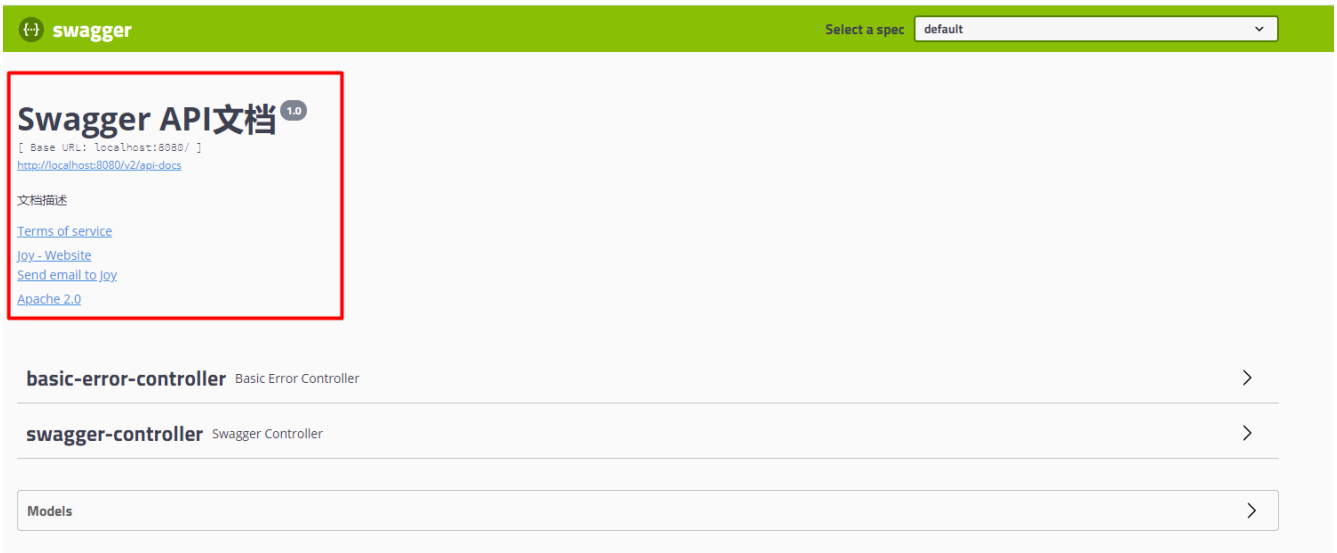
2.5、配置Swagger

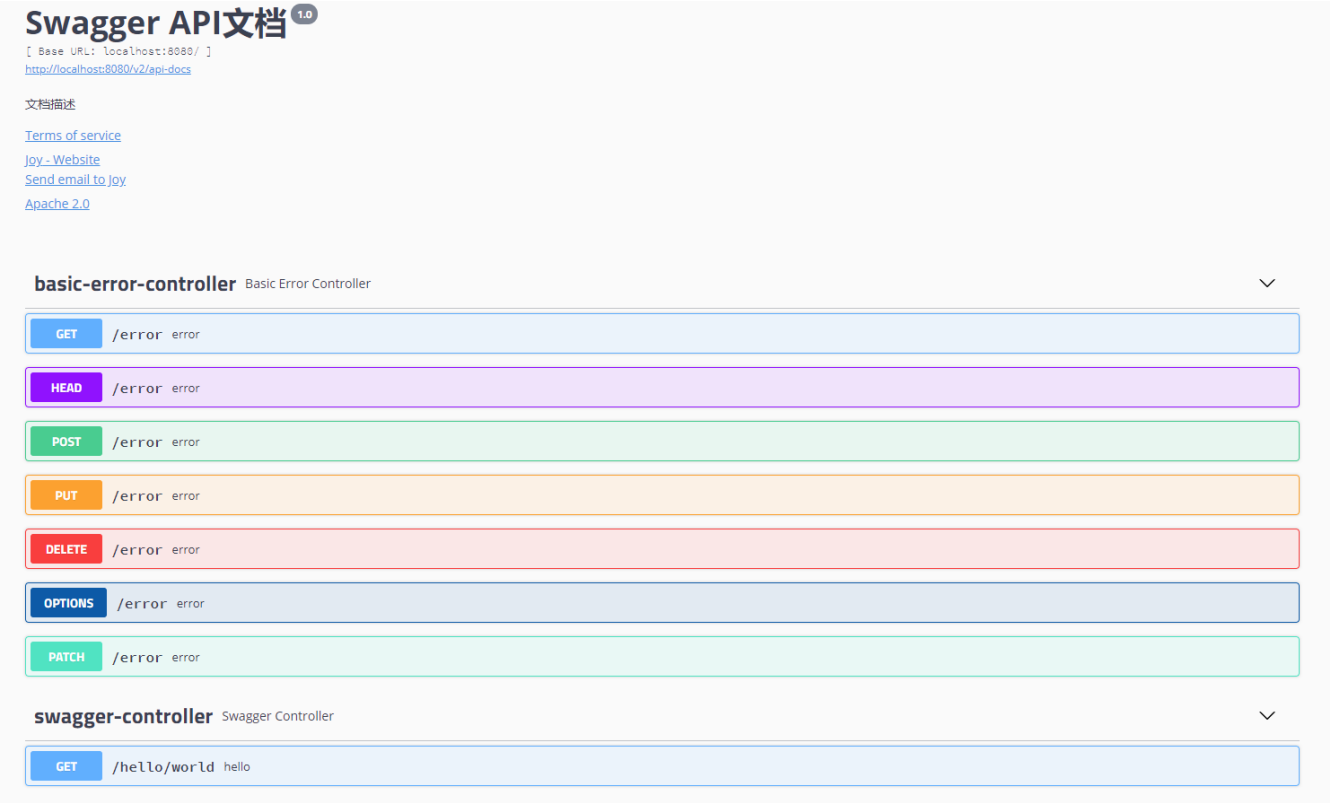
配置Swagger的bean实例：Docket

```java
package com.swagger.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import springfox.documentation.service.ApiInfo;
import springfox.documentation.service.Contact;
import springfox.documentation.spi.DocumentationType;
import springfox.documentation.spring.web.plugins.Docket;
import springfox.documentation.swagger2.annotations.EnableSwagger2;

import java.util.ArrayList;

@Configuration
@EnableSwagger2
```

```java
public class SwaggerConfig {

    //配置Swagger的Docket的bean实例
    @Bean
    public Docket docket() {
        return new Docket(DocumentationType.SWAGGER_2).apiInfo(apiInfo());
    }

    private ApiInfo apiInfo() {
        //作者信息
        Contact contact = new Contact("Joy", "ddd", "111@qq.com");
        return new ApiInfo(
                "Swagger API文档",
                "文档描述",
                "1.0",
                "http://localhost:8080/hello/world",
                contact,
                "Apache 2.0",
                "http://www.apache.org/licenses/LICENSE-2.0",
                new ArrayList());

    }


}
```

效果如下：





2.6、Swagger配置扫描接口

在配置类中添加

```
package com.swagger.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;
import springfox.documentation.RequestHandler;
import springfox.documentation.builders.PathSelectors;
import springfox.documentation.builders.RequestHandlerSelectors;
import springfox.documentation.service.ApiInfo;
import springfox.documentation.service.Contact;
import springfox.documentation.spi.DocumentationType;
import springfox.documentation.spring.web.plugins.Docket;
import springfox.documentation.swagger2.annotations.EnableSwagger2;

import java.util.ArrayList;

@Configuration
@EnableSwagger2
public class SwaggerConfig {

    //配置Swagger的Docket的bean实例
    @Bean
```

```
    public Docket docket() {
        return new Docket(DocumentationType.SWAGGER_2)
                .apiInfo(apiInfo())
                .select()
                //RequestHandlerSelectors:配置要扫描的接口的方式
                //basePackage:指定要扫描的包
                //any():扫描全部的包
                //none():不扫描包
                //withClassAnnotation():扫描类上的注解，传入注解的反射对象
//                .apis(RequestHandlerSelectors.withClassAnnotation(RestController.class))
                //withMethodAnnotation():扫描方法上的注解
//                .apis(RequestHandlerSelectors.withMethodAnnotation(GetMapping.class))
                .apis(RequestHandlerSelectors.basePackage("com.swagger.controller"))
                //paths():用于过滤接口路径
                .paths(PathSelectors.ant("/hello/**"))
                .build();
    }

    private ApiInfo apiInfo() {
        //作者信息
        Contact contact = new Contact("Joy", "ddd", "111@qq.com");
        return new ApiInfo(
                "Swagger API文档",
                "文档描述",
                "1.0",
                "http://localhost:8080/hello/world",
                contact,
                "Apache 2.0",
                "http://www.apache.org/licenses/LICENSE-2.0",
                new ArrayList());

    }


}
```

2.7、配置swagger的启动与关闭

Docket中的enable方法可以控制swagger的启动与关闭，设置为false，启动将无法访问到API页面

> enable(boolean isOpen)

```
package com.swagger.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;
```

```java
import springfox.documentation.RequestHandler;
import springfox.documentation.builders.PathSelectors;
import springfox.documentation.builders.RequestHandlerSelectors;
import springfox.documentation.service.ApiInfo;
import springfox.documentation.service.Contact;
import springfox.documentation.spi.DocumentationType;
import springfox.documentation.spring.web.plugins.Docket;
import springfox.documentation.swagger2.annotations.EnableSwagger2;

import java.util.ArrayList;

@Configuration
@EnableSwagger2
public class SwaggerConfig {

    //配置Swagger的Docket的bean实例
    @Bean
    public Docket docket() {
        return new Docket(DocumentationType.SWAGGER_2)
                .apiInfo(apiInfo())
                .enable(false)//设置swagger关闭
                .select()
                .apis(RequestHandlerSelectors.basePackage("com.swagger.controller"))
                //paths():用于过滤接口路径
                .paths(PathSelectors.ant("/hello/**"))
                .build();
    }

    private ApiInfo apiInfo() {
        //作者信息
        Contact contact = new Contact("Joy", "ddd", "111@qq.com");
        return new ApiInfo(
                "Swagger API文档",
                "文档描述",
                "1.0",
                "http://localhost:8080/hello/world",
                contact,
                "Apache 2.0",
                "http://www.apache.org/licenses/LICENSE-2.0",
                new ArrayList());

    }


}
```

2.8、设置在生产环境中使用Swagger

2.8.1、springboot项目设置多配置文件

将原来的配置文件复制两份，分别命名：

> application-dev.properties
>
> application-pro.properties
>
> application.properties

application-dev.properties

```
server.port=8080
```

application-pro.properties

```
server.port=8888
```

application.properties

```
spring.profiles.active=dev
```

这样Springboot启动默认使用application-dev.properties里的配置

2.8.2、设置swagger只在生产环境中使用

在swagger配置文件中获取现在使用的是生产环境的配置还是部署环境的配置

```
package com.swagger.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.core.env.Environment;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;
import springfox.documentation.RequestHandler;
import springfox.documentation.builders.PathSelectors;
import springfox.documentation.builders.RequestHandlerSelectors;
import springfox.documentation.service.ApiInfo;
import springfox.documentation.service.Contact;
import springfox.documentation.spi.DocumentationType;
import springfox.documentation.spring.web.plugins.Docket;
import springfox.documentation.swagger2.annotations.EnableSwagger2;

import java.util.ArrayList;

@Configuration
@EnableSwagger2
public class SwaggerConfig {

    //配置Swagger的Docket的bean实例
    @Bean
    public Docket docket(Environment environment) {

        boolean dev = environment.acceptsProfiles("dev");

        return new Docket(DocumentationType.SWAGGER_2)
                .apiInfo(apiInfo())
```

```
                .enable(dev)//设置swagger关闭
                .select()
                .apis(RequestHandlerSelectors.basePackage("com.swagger.controller"))
                //paths():用于过滤接口路径
                .paths(PathSelectors.ant("/hello/**"))
                .build();
    }

    private ApiInfo apiInfo() {
        //作者信息
        Contact contact = new Contact("Joy", "ddd", "111@qq.com");
        return new ApiInfo(
                "Swagger API文档",
                "文档描述",
                "1.0",
                "http://localhost:8080/hello/world",
                contact,
                "Apache 2.0",
                "http://www.apache.org/licenses/LICENSE-2.0",
                new ArrayList());

    }


}
```

发现当application.properties里配置dev时，swagger开启，配置pro时，swagger关闭

2.9、配置API文档分组

Docket的groupName方法可以为当前的API分一个组，这样不同的开发人员可以定义自己的组，一个组下可以包含多个API

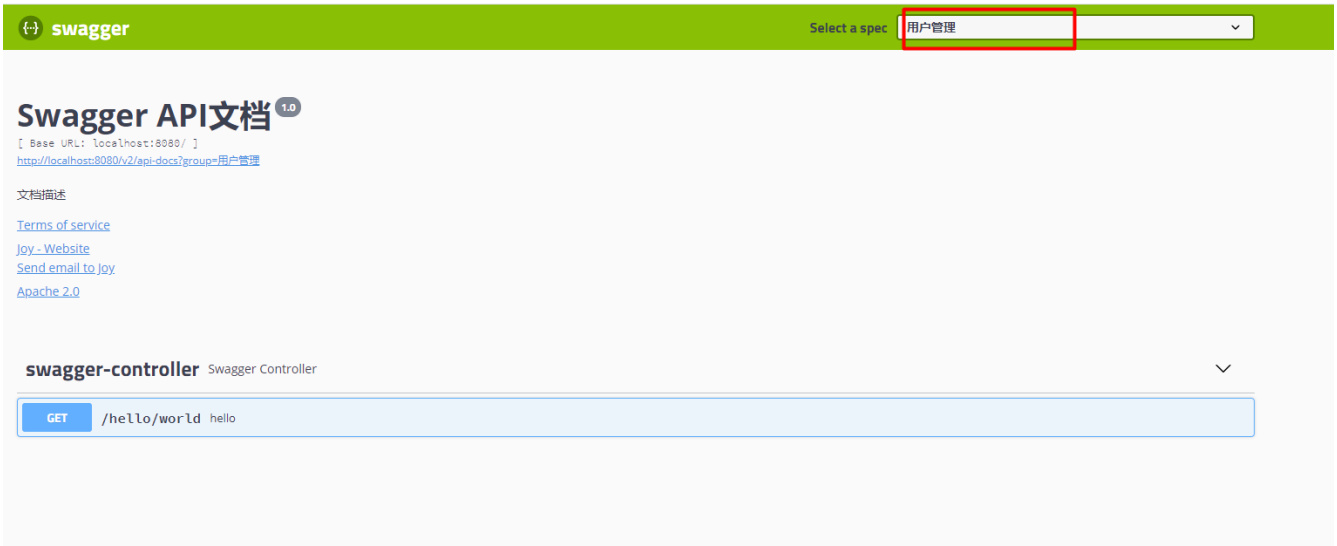
> groupName("用户管理")

Swagger配置类

```
package com.swagger.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.core.env.Environment;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;
import springfox.documentation.RequestHandler;
import springfox.documentation.builders.PathSelectors;
import springfox.documentation.builders.RequestHandlerSelectors;
```

```java
import springfox.documentation.service.ApiInfo;
import springfox.documentation.service.Contact;
import springfox.documentation.spi.DocumentationType;
import springfox.documentation.spring.web.plugins.Docket;
import springfox.documentation.swagger2.annotations.EnableSwagger2;

import java.util.ArrayList;

@Configuration
@EnableSwagger2
public class SwaggerConfig {

    //配置Swagger的Docket的bean实例
    @Bean
    public Docket docket(Environment environment) {

        boolean dev = environment.acceptsProfiles("dev");

        return new Docket(DocumentationType.SWAGGER_2)
                .apiInfo(apiInfo())
                .groupName("用户管理")
                .enable(dev)//设置swagger关闭
                .select()
                .apis(RequestHandlerSelectors.basePackage("com.swagger.controller"))
                //paths():用于过滤接口路径
                .paths(PathSelectors.ant("/hello/**"))
                .build();
    }

    private ApiInfo apiInfo() {
        //作者信息
        Contact contact = new Contact("Joy", "ddd", "111@qq.com");
        return new ApiInfo(
                "Swagger API文档",
                "文档描述",
                "1.0",
                "http://localhost:8080/hello/world",
                contact,
                "Apache 2.0",
                "http://www.apache.org/licenses/LICENSE-2.0",
                new ArrayList());

    }


}
```

效果:



注意：如何配置多个组?

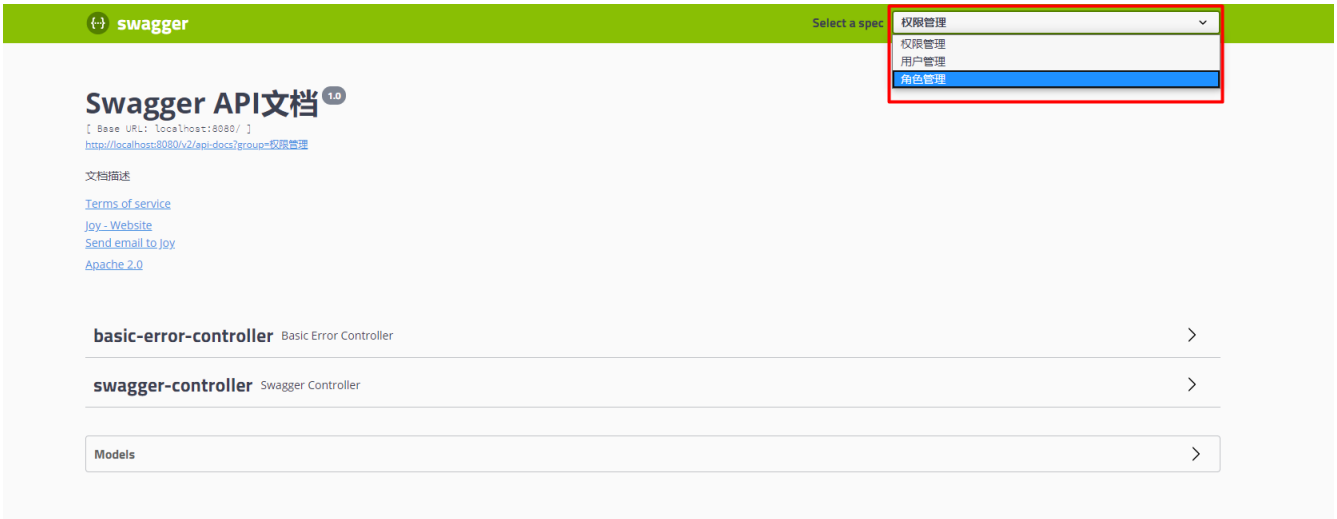
答：可以定义多个Docket配置的bean

```
package com.swagger.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.core.env.Environment;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;
import springfox.documentation.RequestHandler;
import springfox.documentation.builders.PathSelectors;
import springfox.documentation.builders.RequestHandlerSelectors;
import springfox.documentation.service.ApiInfo;
import springfox.documentation.service.Contact;
import springfox.documentation.spi.DocumentationType;
import springfox.documentation.spring.web.plugins.Docket;
import springfox.documentation.swagger2.annotations.EnableSwagger2;

import java.util.ArrayList;

@Configuration
@EnableSwagger2
public class SwaggerConfig {

    @Bean
    public Docket docket2(Environment environment) {

        boolean dev = environment.acceptsProfiles("dev");

        return new Docket(DocumentationType.SWAGGER_2)
                .apiInfo(apiInfo())
                .groupName("角色管理");
    }
```

```java
    @Bean
    public Docket docket3(Environment environment) {

        boolean dev = environment.acceptsProfiles("dev");

        return new Docket(DocumentationType.SWAGGER_2)
                .apiInfo(apiInfo())
                .groupName("权限管理");
    }


    //配置Swagger的Docket的bean实例
    @Bean
    public Docket docket(Environment environment) {

        boolean dev = environment.acceptsProfiles("dev");

        return new Docket(DocumentationType.SWAGGER_2)
                .apiInfo(apiInfo())
                .groupName("用户管理")
                .enable(dev)//设置swagger关闭
                .select()
                //RequestHandlerSelectors:配置要扫描的接口的方式
                //basePackage:指定要扫描的包
                //any():扫描全部的包
                //none():不扫描包
                //withClassAnnotation():扫描类上的注解，传入注解的反射对象
//                .apis(RequestHandlerSelectors.withClassAnnotation(RestController.class))
                //withMethodAnnotation():扫描方法上的注解
//                .apis(RequestHandlerSelectors.withMethodAnnotation(GetMapping.class))
                .apis(RequestHandlerSelectors.basePackage("com.swagger.controller"))
                //paths():用于过滤接口路径
                .paths(PathSelectors.ant("/hello/**"))
                .build();
    }

    private ApiInfo apiInfo() {
        //作者信息
        Contact contact = new Contact("Joy", "ddd", "111@qq.com");
        return new ApiInfo(
                "Swagger API文档",
                "文档描述",
                "1.0",
                "http://localhost:8080/hello/world",
                contact,
                "Apache 2.0",
                "http://www.apache.org/licenses/LICENSE-2.0",
                new ArrayList());

    }
```

```
}
```

效果如下：



不同的开发人员定义不同的组，配置的Docket的bean可以不在一个配置类中，但都会交给Spring管理

2.10、swagger配置实体类

只要接口中返回值中有实体类，就可以被swagger会扫描到swagger中，

演示：

创建实体类：

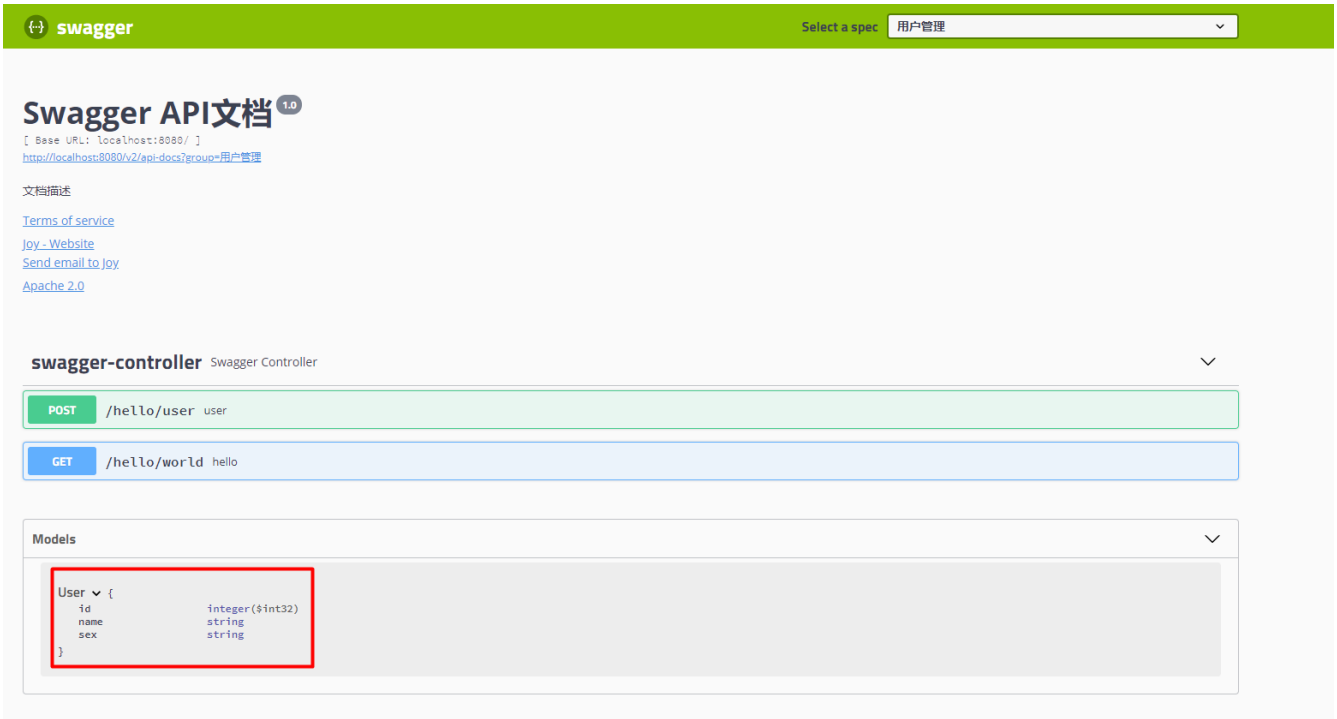
```
package com.swagger.pojo;

import lombok.Data;
import lombok.EqualsAndHashCode;

@Data
@EqualsAndHashCode(callSuper = false)
public class User {

    private Integer id;

    private  String name;

    private  String sex;
```

```
}
```

添加接口：

```
package com.swagger.controller;

import com.swagger.pojo.User;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping("/hello")
public class SwaggerController {

    @GetMapping("/world")
    public String hello(){
        return "你好，世界";
    }

    @PostMapping("/user")
    public User user(){
        return new User();
    }



}
```

效果如下：



- 如何给实体类加注释

```
package com.swagger.pojo;

import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;
import lombok.Data;
import lombok.EqualsAndHashCode;

@Data
@EqualsAndHashCode(callSuper = false)
@ApiModel("用户实体类")
public class User {

    @ApiModelProperty("id")
    private Integer id;

    @ApiModelProperty("名称")
    private  String name;

    @ApiModelProperty("性别")
    private  String sex;




}
```

效果如下：

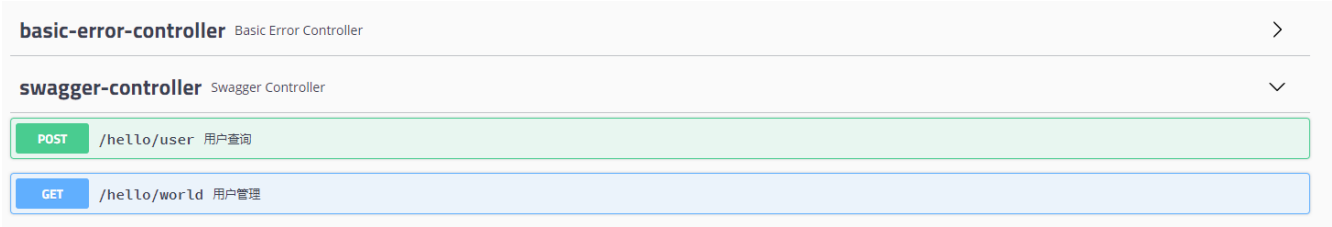
2.11、给接口加中文注释

> @ApiOperation("注释内容“）用在方法中，给接口加注释
>
> @ApiParam("给参数加注释") 用在参数列表中，给参数加注释

Api如下：

```
package com.swagger.controller;

import com.swagger.pojo.User;
import io.swagger.annotations.ApiOperation;
import io.swagger.annotations.ApiParam;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping("/hello")
public class SwaggerController {

    @GetMapping("/world")
    @ApiOperation("用户管理")
    public String hello(){
        return "你好，世界";
    }

    @PostMapping("/user")
    @ApiOperation("用户查询")
    public User user(@ApiParam("用户名字")String userName){
        return new User();
    }



}
```

效果如下：


basic-error-controller Basic Error Controller
swagger-controller Swagger Controller
POST /hello/user 用户查询
GET /hello/world 用户管理