swagger

介绍

相信无论是前端还是后端开发，都或多或少地被接口文档折磨过。前端经常抱怨后端给的接口文档与实际情况不一致。后端又觉得编写及维护接口文档会耗费不少精力，经常来不及更新。其实无论是前端调用后端，还是后端调用后端，都期望有一个好的接口文档。但是这个接口文档对于程序员来说，就跟注释一样，经常会抱怨别人写的代码没有写注释，然而自己写起代码起来，最讨厌的，也是写注释。所以仅仅只通过强制来规范大家是不够的，随着时间推移，版本迭代，接口文档往往很容易就跟不上代码了。

使用Swagger你只需要按照它的规范去定义接口及接口相关的信息。再通过Swagger衍生出来的一系列项目和工具，就可以做到生成各种格式的接口文档，生成多种语言的客户端和服务端的代码，以及在线接口调试页面等等。这样，如果按照新的开发模式，在开发新版本或者迭代版本的时候，只需要更新Swagger描述文件，就可以自动生成接口文档和客户端服务端代码，做到调用端代码、服务端代码以及接口文档的一致性。

为了简化swagger的使用，Spring框架对swagger进行了整合，建立了Spring-swagger项目，后面改成了现在的Springfox。通过在项目中引入Springfox，可以扫描相关的代码，生成描述文件，进而生成与代码一致的接口文档和客户端代码。

Springfox对应的maven坐标如下：

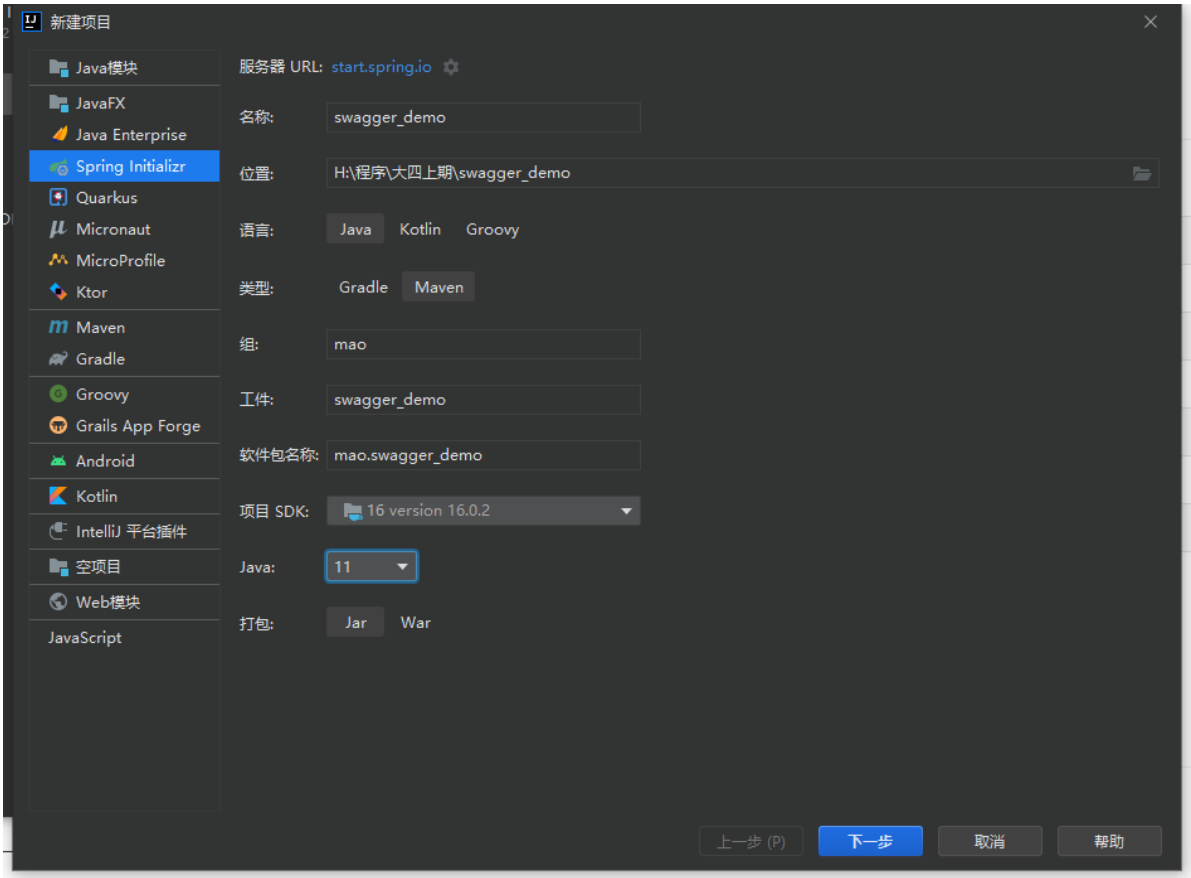
```
<dependency>
    <groupId>io.springfox</groupId>
    <artifactId>springfox-swagger-ui</artifactId>
    <version>2.9.2</version>
</dependency>
<dependency>
    <groupId>io.springfox</groupId>
    <artifactId>springfox-swagger2</artifactId>
    <version>2.9.2</version>
</dependency>
```

swagger常用注解

注解	说明
@Api	用在请求的类上，例如Controller，表示对类的说明
@ApiModel	用在类上，通常是实体类，表示一个返回响应数据的信息
@ApiModelProperty	用在属性上，描述响应类的属性
@ApiOperation	用在请求的方法上，说明方法的用途、作用
@ApiImplicitParams	用在请求的方法上，表示一组参数说明
@ApiImplicitParam	用在@ApiImplicitParams注解中，指定一个请求参数的各个方面

swagger入门案例

第一步：创建工程swagger_demo



第二步：修改pom文件

```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>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.7.1</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>mao</groupId>
    <artifactId>swagger_demo</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>swagger_demo</name>
    <description>swagger_demo</description>
    <properties>
        <java.version>11</java.version>
    </properties>
    <dependencies>
        <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>
        </dependency>

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

    </dependencies>

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

```
        </plugins>
    </build>

</project>

```

第三步：创建实体类User

```
package mao.swagger_demo.entity;

/**
 * Project name(项目名称)：swagger_demo
 * Package(包名): mao.swagger_demo.entity
 * Class(类名): User
 * Author(作者）: mao
 * Author QQ：1296193245
 * GitHub：https://github.com/maomao124/
 * Date(创建日期)： 2022/10/26
 * Time(创建时间)： 20:19
 * Version(版本): 1.0
 * Description(描述)： 无
 */


public class User
{
    /**
     * id
     */
    private long id;

    /**
     * 名字
     */
    private String name;

    /**
     * 密码
     */
    private String password;

}
```

第四步：给实体类User添加对应的注解

```java
package mao.swagger_demo.entity;

import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;

/**
 * Project name(项目名称)：swagger_demo
 * Package(包名): mao.swagger_demo.entity
 * Class(类名): User
 * Author(作者）: mao
 * Author QQ：1296193245
 * GitHub：https://github.com/maomao124/
 * Date(创建日期)： 2022/10/26
 * Time(创建时间)： 20:19
 * Version(版本): 1.0
 * Description(描述)： 无
 */

@ApiModel(description = "用户实体类")
public class User
{
    /**
     * id
     */
    @ApiModelProperty(value = "id，主键")
    private long id;

    /**
     * 名字
     */
    @ApiModelProperty(value = "用户的名字或者昵称")
    private String name;

    /**
     * 密码
     */
    @ApiModelProperty(value = "用户的密码")
    private String password;


    public User()
    {
    }

    public User(long id, String name, String password)
    {
        this.id = id;
        this.name = name;
        this.password = password;
    }

    public long getId()
    {
```

```
        return id;
    }

    public void setId(long id)
    {
        this.id = id;
    }

    public String getName()
    {
        return name;
    }

    public void setName(String name)
    {
        this.name = name;
    }

    public String getPassword()
    {
        return password;
    }

    public void setPassword(String password)
    {
        this.password = password;
    }

    @Override
    @SuppressWarnings("all")
    public String toString()
    {
        final StringBuilder stringbuilder = new StringBuilder();
        stringbuilder.append("id: ").append(id).append('\n');
        stringbuilder.append("name: ").append(name).append('\n');
        stringbuilder.append("password: ").append(password).append('\n');
        return stringbuilder.toString();
    }
}
```

第五步：创建实体类Menu

```
package mao.swagger_demo.entity;

/**
 * Project name(项目名称)：swagger_demo
 * Package(包名): mao.swagger_demo.entity
 * Class(类名): Menu
 * Author(作者）: mao
 * Author QQ：1296193245
```

```
 * GitHub：https://github.com/maomao124/
 * Date(创建日期)：  2022/10/26
 * Time(创建时间)：  20:34
 * Version(版本)：1.0
 * Description(描述)：  无
 */

public class Menu
{
    /**
     * id
     */
    private long id;

    /**
     * 名字
     */
    private String name;

    /**
     * 类型
     */
    private int type;

    /**
     * 子菜单
     */
    private Menu subMenu;
}
```

第六步：给实体类Menu添加对应的注解

```
package mao.swagger_demo.entity;

import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;

/**
 * Project name(项目名称)：swagger_demo
 * Package(包名)：mao.swagger_demo.entity
 * Class(类名)：Menu
 * Author(作者）：mao
 * Author QQ：1296193245
 * GitHub：https://github.com/maomao124/
 * Date(创建日期)：  2022/10/26
 * Time(创建时间)：  20:34
 * Version(版本)：1.0
 * Description(描述)：  无
 */

@ApiModel(description = "菜单实体类")
```

```java
public class Menu
{
    /**
     * id
     */
    @ApiModelProperty(value = "主键")
    private long id;

    /**
     * 名字
     */
    @ApiModelProperty(value = "菜单名称")
    private String name;

    /**
     * 类型
     */
    @ApiModelProperty(value = "菜单的类型，如果为0，则为有子菜单的菜单，如果为1，则为没有子菜单的菜单项")
    private int type;

    /**
     * 子菜单
     */
    @ApiModelProperty(value = "子菜单，如果当前为菜单项，则此字段的值为null")
    private Menu subMenu;

    /**
     * Instantiates a new Menu.
     */
    public Menu()
    {
    }

    /**
     * Instantiates a new Menu.
     *
     * @param id      the id
     * @param name    the name
     * @param type    the type
     * @param subMenu the sub menu
     */
    public Menu(long id, String name, int type, Menu subMenu)
    {
        this.id = id;
        this.name = name;
        this.type = type;
        this.subMenu = subMenu;
    }

    /**
     * Gets id.
     *
     * @return the id
     */
    public long getId()
    {
        return id;
```

```
77      }
78
79      /**
80       * Sets id.
81       *
82       * @param id the id
83       */
84      public void setId(long id)
85      {
86          this.id = id;
87      }
88
89      /**
90       * Gets name.
91       *
92       * @return the name
93       */
94      public String getName()
95      {
96          return name;
97      }
98
99      /**
100      * Sets name.
101      *
102      * @param name the name
103      */
104     public void setName(String name)
105     {
106         this.name = name;
107     }
108
109     /**
110      * Gets type.
111      *
112      * @return the type
113      */
114     public int getType()
115     {
116         return type;
117     }
118
119     /**
120      * Sets type.
121      *
122      * @param type the type
123      */
124     public void setType(int type)
125     {
126         this.type = type;
127     }
128
129     /**
130      * Gets sub menu.
131      *
132      * @return the sub menu
133      */
134     public Menu getSubMenu()
```

```
    {
        return subMenu;
    }

    /**
     * Sets sub menu.
     *
     * @param subMenu the sub menu
     */
    public void setSubMenu(Menu subMenu)
    {
        this.subMenu = subMenu;
    }

    @Override
    @SuppressWarnings("all")
    public String toString()
    {
        final StringBuilder stringbuilder = new StringBuilder();
        stringbuilder.append("id: ").append(id).append('\n');
        stringbuilder.append("name: ").append(name).append('\n');
        stringbuilder.append("type: ").append(type).append('\n');
        stringbuilder.append("subMenu: ").append(subMenu).append('\n');
        return stringbuilder.toString();
    }
}
```

第七步：创建实体类Student

```
package mao.swagger_demo.entity;

/**
 * Project name(项目名称)：swagger_demo
 * Package(包名): mao.swagger_demo.entity
 * Class(类名): Student
 * Author(作者）: mao
 * Author QQ：1296193245
 * GitHub：https://github.com/maomao124/
 * Date(创建日期)： 2022/10/26
 * Time(创建时间)： 20:41
 * Version(版本): 1.0
 * Description(描述)： 无
 */

public class Student
{
    /**
     * id
     */
    private long id;
    /**
```

```
     * 名字
     */
    private String name;
    /**
     * 性
     */
    private String sex;
    /**
     * 年龄
     */
    private int age;
}
```

第八步：给实体类Student添加对应的注解

```
package mao.swagger_demo.entity;

import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;

/**
 * Project name(项目名称)：swagger_demo
 * Package(包名): mao.swagger_demo.entity
 * Class(类名): Student
 * Author(作者）: mao
 * Author QQ：1296193245
 * GitHub：https://github.com/maomao124/
 * Date(创建日期)： 2022/10/26
 * Time(创建时间)： 20:41
 * Version(版本): 1.0
 * Description(描述)： 无
 */


@ApiModel(description = "学生实体类")
public class Student
{
    /**
     * id
     */
    @ApiModelProperty(value = "学生的学号，主键")
    private long id;
    /**
     * 名字
     */
    @ApiModelProperty(value = "学生姓名")
    private String name;
    /**
     * 性
     */
    @ApiModelProperty(value = "学生性别")
```

```
37      private String sex;
38      /**
39       * 年龄
40       */
41      @ApiModelProperty(value = "学生的年龄")
42      private int age;
43
44      /**
45       * Instantiates a new Student.
46       */
47      public Student()
48      {
49      }
50
51      /**
52       * Instantiates a new Student.
53       *
54       * @param id   the id
55       * @param name the name
56       * @param sex  the sex
57       * @param age  the age
58       */
59      public Student(long id, String name, String sex, int age)
60      {
61          this.id = id;
62          this.name = name;
63          this.sex = sex;
64          this.age = age;
65      }
66
67      /**
68       * Gets id.
69       *
70       * @return the id
71       */
72      public long getId()
73      {
74          return id;
75      }
76
77      /**
78       * Sets id.
79       *
80       * @param id the id
81       */
82      public void setId(long id)
83      {
84          this.id = id;
85      }
86
87      /**
88       * Gets name.
89       *
90       * @return the name
91       */
92      public String getName()
93      {
94          return name;
```

```
95      }
96
97      /**
98       * Sets name.
99       *
100      * @param name the name
101      */
102     public void setName(String name)
103     {
104         this.name = name;
105     }
106
107     /**
108      * Gets sex.
109      *
110      * @return the sex
111      */
112     public String getSex()
113     {
114         return sex;
115     }
116
117     /**
118      * Sets sex.
119      *
120      * @param sex the sex
121      */
122     public void setSex(String sex)
123     {
124         this.sex = sex;
125     }
126
127     /**
128      * Gets age.
129      *
130      * @return the age
131      */
132     public int getAge()
133     {
134         return age;
135     }
136
137     /**
138      * Sets age.
139      *
140      * @param age the age
141      */
142     public void setAge(int age)
143     {
144         this.age = age;
145     }
146
147     @Override
148     @SuppressWarnings("all")
149     public String toString()
150     {
151         final StringBuilder stringbuilder = new StringBuilder();
152         stringbuilder.append("id: ").append(id).append('\n');
```

```
        stringbuilder.append("name: ").append(name).append('\n');
        stringbuilder.append("sex: ").append(sex).append('\n');
        stringbuilder.append("age: ").append(age).append('\n');
        return stringbuilder.toString();
    }
}
```

第九步：创建UserController

```
package mao.swagger_demo.controller;

import mao.swagger_demo.entity.User;
import org.springframework.web.bind.annotation.*;

import java.util.ArrayList;
import java.util.List;

/**
 * Project name(项目名称)：swagger_demo
 * Package(包名): mao.swagger_demo.controller
 * Class(类名): UserController
 * Author(作者）: mao
 * Author QQ：1296193245
 * GitHub：https://github.com/maomao124/
 * Date(创建日期)： 2022/10/26
 * Time(创建时间)： 20:45
 * Version(版本): 1.0
 * Description(描述)： 无
 */

@RestController
@RequestMapping("/user")
public class UserController
{
    @GetMapping("/{id}")
    public User getUser(@PathVariable long id)
    {
        return new User(id, "张三", "1234");
    }

    @GetMapping
    public List<User> getAll()
    {
        List<User> list = new ArrayList<>(3);
        list.add(new User(1, "张三", "1234"));
        list.add(new User(2, "张三", "1234"));
        list.add(new User(3, "张三", "1234"));
        return list;
    }

```

```
    @PostMapping("")
    public String saveUser()
    {
        return "ok";
    }

    @PutMapping
    public String updateUser()
    {
        return "ok";
    }

    @DeleteMapping
    public String deleteUser()
    {
        return "ok";
    }
}
```

第十步：给UserController添加对应的注解

```
package mao.swagger_demo.controller;

import io.swagger.annotations.Api;
import io.swagger.annotations.ApiImplicitParam;
import io.swagger.annotations.ApiImplicitParams;
import io.swagger.annotations.ApiOperation;
import mao.swagger_demo.entity.User;
import org.springframework.web.bind.annotation.*;

import java.util.ArrayList;
import java.util.List;

/**
 * Project name(项目名称)：swagger_demo
 * Package(包名): mao.swagger_demo.controller
 * Class(类名): UserController
 * Author(作者）: mao
 * Author QQ：1296193245
 * GitHub：https://github.com/maomao124/
 * Date(创建日期)： 2022/10/26
 * Time(创建时间)： 20:45
 * Version(版本): 1.0
 * Description(描述)： 无
 */

@Api(tags = "用户控制器")
@RestController
@RequestMapping("/user")
public class UserController
{
```

```
    @GetMapping("/{id}")
    @ApiOperation(value = "根据用户id查询用户", notes = "根据用户id查询用户")
    @ApiImplicitParams
            (
                    @ApiImplicitParam(name = "id", value = "用户的id",
required = true, type = "long")
            )
    public User getUser(@PathVariable long id)
    {
        return new User(id, "张三", "1234");
    }

    @GetMapping
    @ApiOperation(value = "查询所有用户", notes = "查询所有用户的信息")
    public List<User> getAll()
    {
        List<User> list = new ArrayList<>(3);
        list.add(new User(1, "张三", "1234"));
        list.add(new User(2, "张三", "1234"));
        list.add(new User(3, "张三", "1234"));
        return list;
    }

    @PostMapping("")
    @ApiOperation(value = "保存(添加)用户的信息", notes = "保存(添加)用户的信息")
    public String saveUser()
    {
        return "ok";
    }

    @PutMapping
    @ApiOperation(value = "更新用户信息", notes = "更新用户信息")
    public String updateUser()
    {
        return "ok";
    }

    @DeleteMapping
    @ApiOperation(value = "删除用户的信息", notes = "删除用户的信息")
    public String deleteUser()
    {
        return "ok";
    }
}
```

第十一步：创建MenuController

```
package mao.swagger_demo.controller;

import org.springframework.web.bind.annotation.*;
```

```
4
5  /**
6   * Project name(项目名称)：swagger_demo
7   * Package(包名): mao.swagger_demo.controller
8   * Class(类名): MenuController
9   * Author(作者）: mao
10  * Author QQ：1296193245
11  * GitHub：https://github.com/maomao124/
12  * Date(创建日期)： 2022/10/26
13  * Time(创建时间)： 21:02
14  * Version(版本): 1.0
15  * Description(描述)： 无
16  */
17
18
19  public class MenuController
20  {
21      @PostMapping("/save")
22      public String save()
23      {
24          return "OK";
25      }
26
27      @PutMapping("/update")
28      public String update()
29      {
30          return "OK";
31      }
32
33      @DeleteMapping("/delete")
34      public String delete(int id)
35      {
36          return "OK";
37      }
38
39
40      @GetMapping(value = "page/{pageNum}/{pageSize}")
41      public List<Menu> findByPage(@PathVariable Integer pageNum,
42                                   @PathVariable Integer pageSize)
43      {
44          return null;
45      }
46  }
```

第十二步：给MenuController添加相应的注解

```
1  package mao.swagger_demo.controller;
2
3  import io.swagger.annotations.Api;
4  import io.swagger.annotations.ApiImplicitParam;
5  import io.swagger.annotations.ApiImplicitParams;
```

```java
import io.swagger.annotations.ApiOperation;
import org.springframework.web.bind.annotation.*;

/**
 * Project name(项目名称)：swagger_demo
 * Package(包名): mao.swagger_demo.controller
 * Class(类名): MenuController
 * Author(作者）: mao
 * Author QQ：1296193245
 * GitHub：https://github.com/maomao124/
 * Date(创建日期)： 2022/10/26
 * Time(创建时间)： 21:02
 * Version(版本): 1.0
 * Description(描述)： 无
 */


@RestController()
@RequestMapping("/menu")
@Api(tags = "菜单控制器")
public class MenuController
{
    @PostMapping("/save")
    @ApiOperation(value = "添加菜单", notes = "添加菜单")
    public String save()
    {
        return "OK";
    }

    @PutMapping("/update")
    @ApiOperation(value = "更新菜单", notes = "更新菜单")
    public String update()
    {
        return "OK";
    }

    @DeleteMapping("/delete")
    @ApiOperation(value = "删除菜单", notes = "删除菜单")
    @ApiImplicitParams
            (
                    @ApiImplicitParam(name = "id", value = "菜单的id", required = true, type = "long")
            )
    public String delete(long id)
    {
        return "OK";
    }


    @GetMapping(value = "page/{pageNum}/{pageSize}")
    @ApiOperation(value = "分页查询", notes = "分页查询菜单信息")
    @ApiImplicitParams
            (
                    {
                            @ApiImplicitParam(name = "pageNum", value = "页码", required = true, type = "Integer"),
                            @ApiImplicitParam(name = "pageSize", value = "每页能显示的条数", required = true, type = "Integer")
```

```
                    }
            )
    public List<Menu> findByPage(@PathVariable Integer pageNum,
                                 @PathVariable Integer pageSize)
    {
        return null;
    }
}

```

第十三步：创建StudentController

```
package mao.swagger_demo.controller;

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

/**
 * Project name(项目名称)：swagger_demo
 * Package(包名): mao.swagger_demo.controller
 * Class(类名): StudentController
 * Author(作者）: mao
 * Author QQ：1296193245
 * GitHub：https://github.com/maomao124/
 * Date(创建日期)： 2022/10/26
 * Time(创建时间)： 21:12
 * Version(版本): 1.0
 * Description(描述)： 无
 */

@RestController
@RequestMapping("/student")
public class StudentController
{
    @GetMapping("/{pageNum}/{pageSize}")
    public List<Student> findByPage(@PathVariable Integer pageNum,
                                    @PathVariable Integer pageSize)
    {
        return null;
    }
}
```

第十四步：给StudentController添加相应的注解

```java
package mao.swagger_demo.controller;

import io.swagger.annotations.Api;
import io.swagger.annotations.ApiImplicitParam;
import io.swagger.annotations.ApiImplicitParams;
import io.swagger.annotations.ApiOperation;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

/**
 * Project name(项目名称)：swagger_demo
 * Package(包名): mao.swagger_demo.controller
 * Class(类名): StudentController
 * Author(作者）: mao
 * Author QQ：1296193245
 * GitHub：https://github.com/maomao124/
 * Date(创建日期)： 2022/10/26
 * Time(创建时间)： 21:12
 * Version(版本): 1.0
 * Description(描述)： 无
 */

@RestController
@RequestMapping("/student")
@Api(tags = "学生控制器")
public class StudentController
{
    @GetMapping("/{pageNum}/{pageSize}")
    @ApiOperation(value = "分页查询学生的信息", notes = "分页查询学生的信息")
    @ApiImplicitParams
            (
                    {
                            @ApiImplicitParam(name = "pageNum", value = "页码", required = true, type = "Integer"),
                            @ApiImplicitParam(name = "pageSize", value = "每页能显示的条数", required = true, type = "Integer")
                    }
            )
    public List<Student> findByPage(@PathVariable Integer pageNum,
                                    @PathVariable Integer pageSize)
    {
        return null;
    }
}
```

第十五步：添加配置类SwaggerConfig

```java
package mao.swagger_demo.config;


import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

import springfox.documentation.builders.ApiInfoBuilder;
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;



/**
 * Project name(项目名称)：swagger_demo
 * Package(包名): mao.swagger_demo.config
 * Class(类名): SwaggerConfig
 * Author(作者）: mao
 * Author QQ：1296193245
 * GitHub：https://github.com/maomao124/
 * Date(创建日期)： 2022/10/26
 * Time(创建时间)： 21:19
 * Version(版本): 1.0
 * Description(描述)： 无
 */

@Configuration
@EnableSwagger2
public class SwaggerConfig
{
    private ApiInfo apiInfo()
    {
        return new ApiInfoBuilder()
                .title("API接口文档")
                .contact(new Contact("mao", "https://github.com/maomao124/",
"1234@qq.com"))
                .version("1.0")
                .description("描述")
                .build();
    }

    @Bean
    public Docket docket()
    {
        Docket docket = new Docket(DocumentationType.SWAGGER_2)
                .apiInfo(apiInfo())
                .groupName("默认组")
                .select()
```

```
52
.apis(RequestHandlerSelectors.basePackage("mao.swagger_demo"))
53                  .build();
54          return docket;
55      }
56  }
57
```

第十六步：更改配置文件

```
1  spring:
2    mvc:
3      pathmatch:
4        matching-strategy: ant_path_matcher
```

第十七步：启动程序

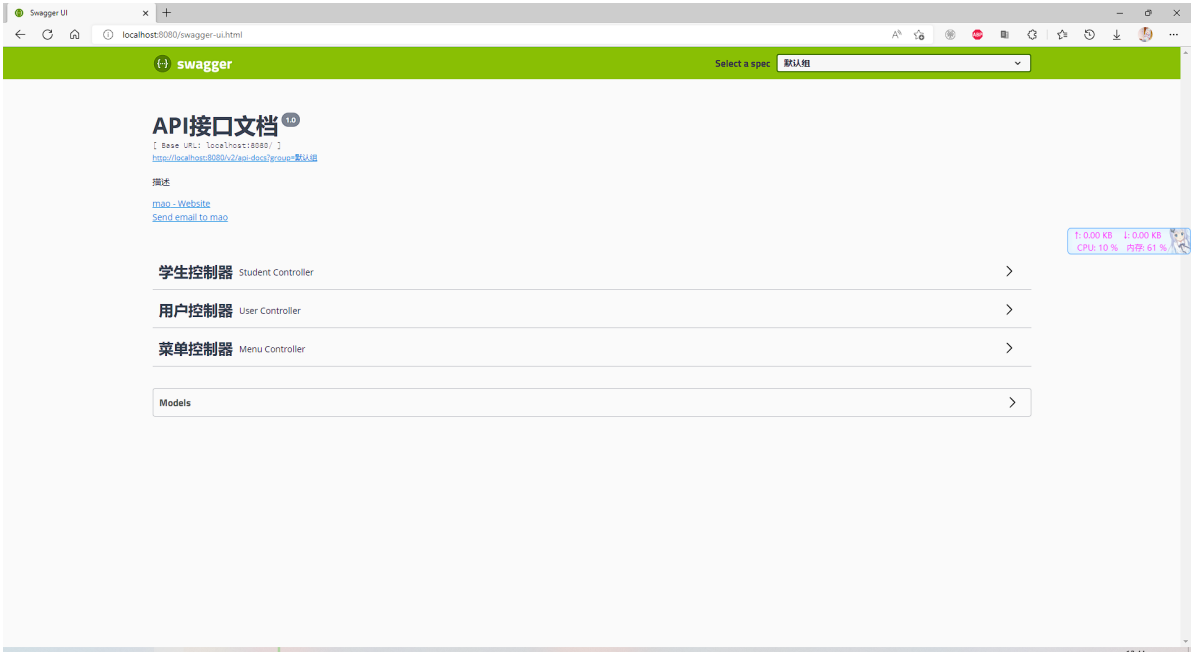
```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
```

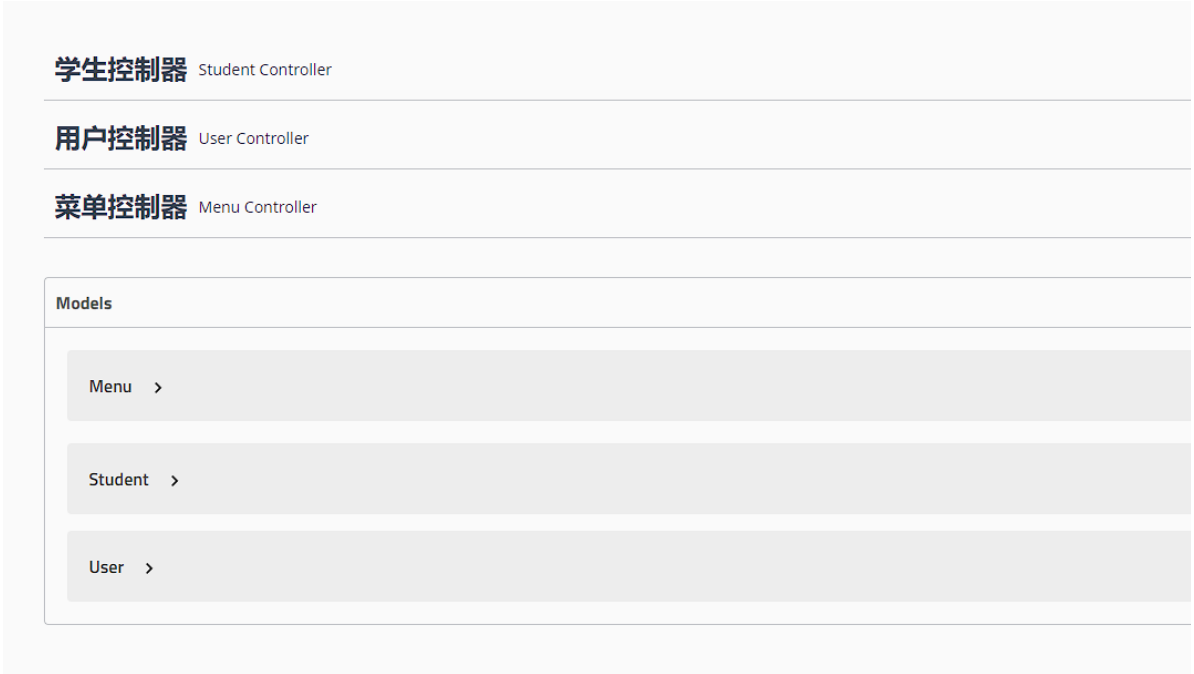
```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v2.7.1)

2022-10-27 12:43:12.425  INFO 15432 --- [           main] mao.swagger_demo.SwaggerDemoApplication  : Starting SwaggerDemoApplication using Java 16.0.2 on mao with PID 15432 (H:\程序\大四上期\swagger_demo\target\classes started by mao in H:\程序\大四上期\swagger_demo)
2022-10-27 12:43:12.427  INFO 15432 --- [           main] mao.swagger_demo.SwaggerDemoApplication  : No active profile set, falling back to 1 default profile: "default"
2022-10-27 12:43:13.257  INFO 15432 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8080 (http)
2022-10-27 12:43:13.264  INFO 15432 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2022-10-27 12:43:13.265  INFO 15432 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Apache Tomcat/9.0.64]
```

```
2022-10-27 12:43:13.350  INFO 15432 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2022-10-27 12:43:13.350  INFO 15432 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 883 ms
2022-10-27 12:43:13.604  INFO 15432 --- [           main] pertySourcedRequestMappingHandlerMapping : Mapped URL path [/v2/api-docs] onto method [springfox.documentation.swagger2.web.Swagger2Controller#getDocumentation(String, HttpServletRequest)]
2022-10-27 12:43:13.753  INFO 15432 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8080 (http) with context path ''
2022-10-27 12:43:13.754  INFO 15432 --- [           main] d.s.w.p.DocumentationPluginsBootstrapper : Context refreshed
2022-10-27 12:43:13.765  INFO 15432 --- [           main] d.s.w.p.DocumentationPluginsBootstrapper : Found 1 custom documentation plugin(s)
2022-10-27 12:43:13.786  INFO 15432 --- [           main] s.d.s.w.s.ApiListingReferenceScanner     : Scanning for api listing references
2022-10-27 12:43:13.882  INFO 15432 --- [           main] .d.s.w.r.o.CachingOperationNameGenerator : Generating unique operation named: findByPageUsingGET_1
2022-10-27 12:43:13.914  INFO 15432 --- [           main] mao.swagger_demo.SwaggerDemoApplication  : Started SwaggerDemoApplication in 1.787 seconds (JVM running for 2.316)
```

第十八步：访问

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


Swagger UI
localhost:8080/swagger-ui.html
swagger
Select a spec
默认组
API接口文档
1.0
[Base URL: localhost:8080/]
http://localhost:8080/v2/api-docs?group=默认组
描述
mao - Website
Send email to mao
学生控制器 Student Controller
用户控制器 User Controller
菜单控制器 Menu Controller
Models


学生控制器 Student Controller
用户控制器 User Controller
菜单控制器 Menu Controller
Models
Menu
Student
User

Models

Menu {

description:	*菜单实体类*
id	integer($int64) **主键**
name	string **菜单名称**

subMenu	{
	description: *子菜单，如果当前为菜单项，则此字段的值为null*
	}

type	integer($int32) **菜单的类型，如果为0，则为有子菜单的菜单，如果为1，则为没有子菜单的菜单项**

}

Student {

description:	*学生实体类*
age	integer($int32) **学生的年龄**
id	integer($int64) **学生的学号，主键**
name	string **学生姓名**
sex	string **学生性别**

}

User {

description:	*用户实体类*
id	integer($int64) **id，主键**
name	string **用户的名字或者昵称**
password	string **用户的密码**

}

学生控制器 Student Controller

GET /student/{pageNum}/{pageSize} 分页查询学生的信息

用户控制器 User Controller

GET /user 查询所有用户

POST /user 保存(添加)用户的信息

PUT /user 更新用户信息

DELETE /user 删除用户的信息

GET /user/{id} 根据用户id查询用户

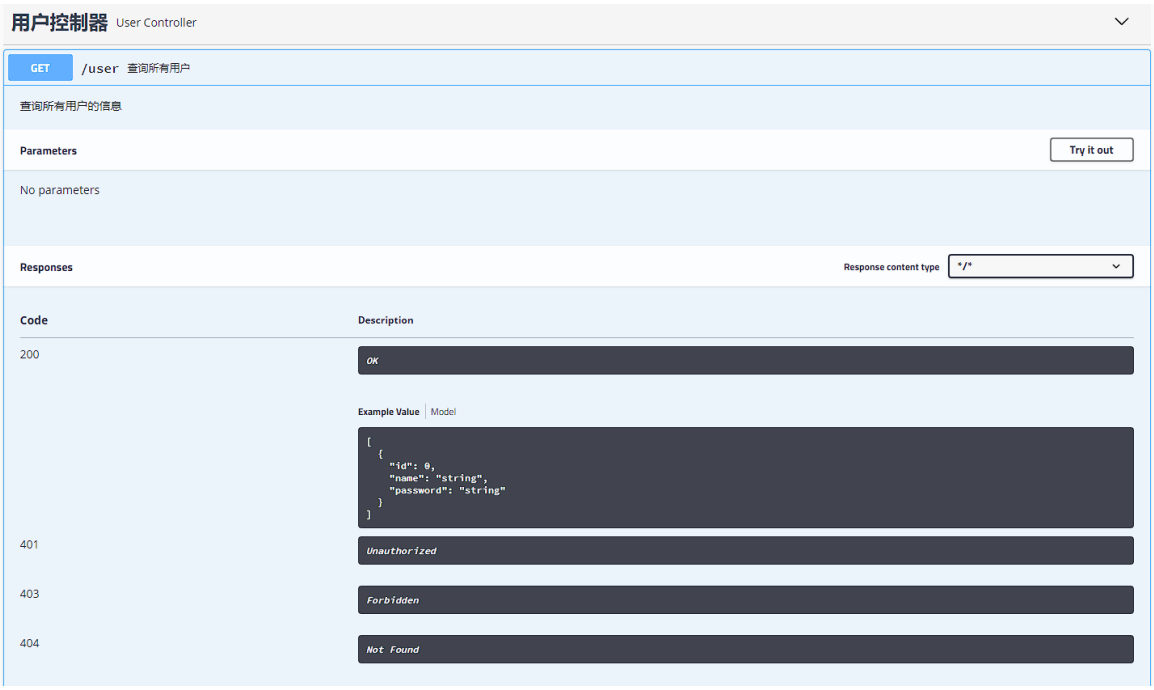
菜单控制器 Menu Controller

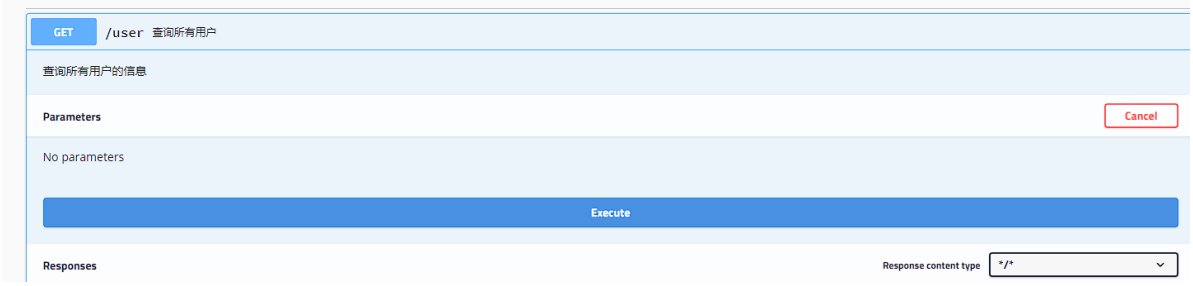
DELETE /menu/delete 删除菜单

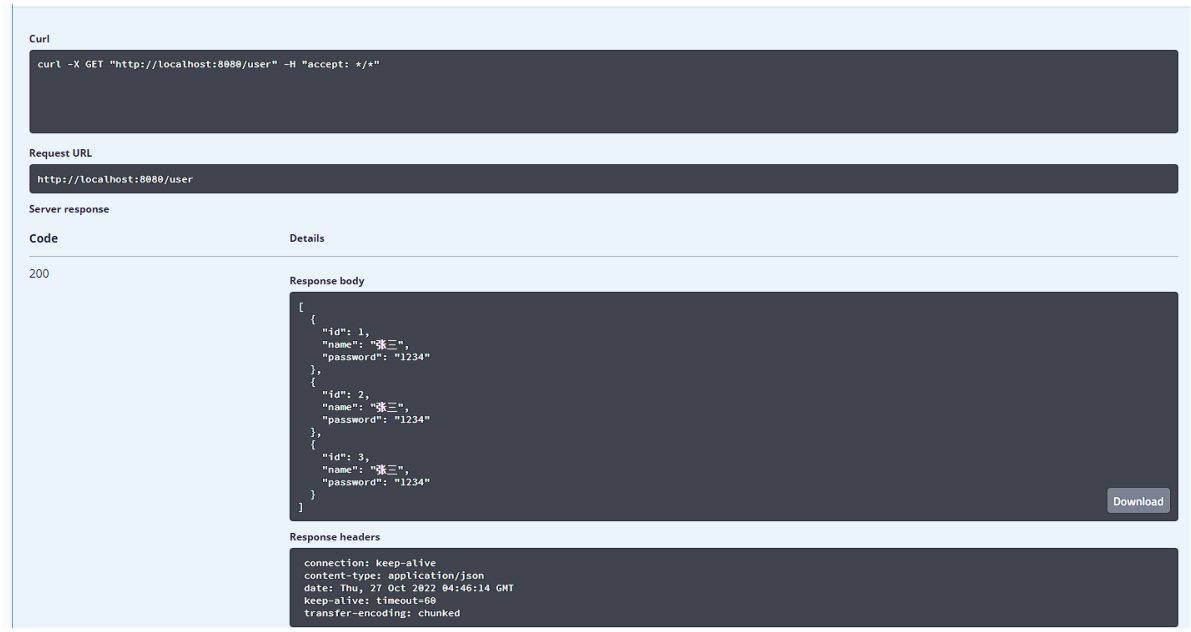
GET /menu/page/{pageNum}/{pageSize} 分页查询

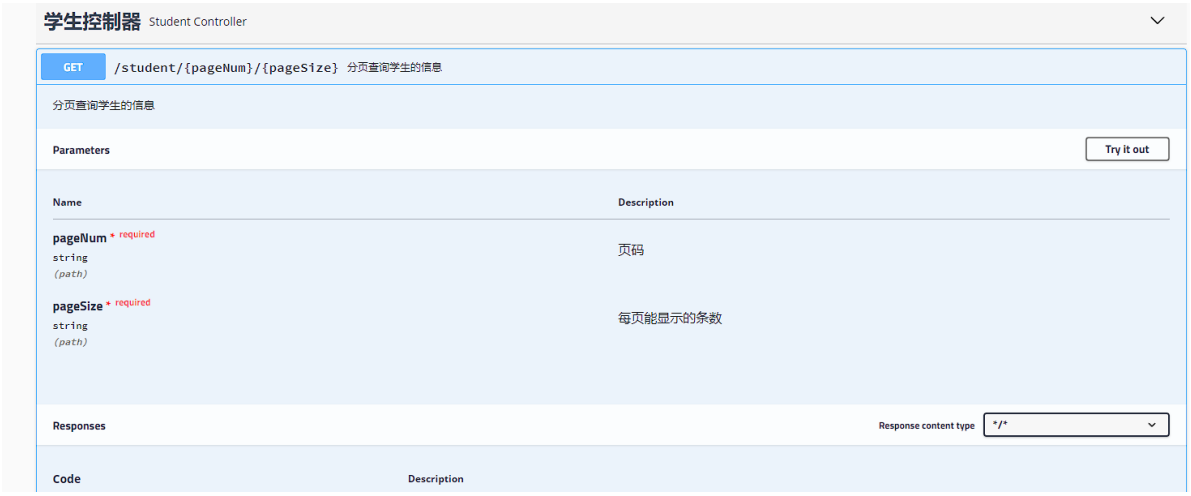
POST /menu/save 添加菜单

PUT /menu/update 更新菜单




GET
/user 查询所有用户
查询所有用户的信息
Parameters
Cancel
No parameters
Execute
Responses
Response content type
/


Curl
curl -X GET "http://localhost:8080/user" -H "accept: */*"
Request URL
http://localhost:8080/user
Server response
Code
Details
200
Response body
[
{
"id": 1,
"name": "张三",
"password": "1234"
},
{
"id": 2,
"name": "张三",
"password": "1234"
},
{
"id": 3,
"name": "张三",
"password": "1234"
}
]
Download
Response headers
connection: keep-alive
content-type: application/json
date: Thu, 27 Oct 2022 04:46:14 GMT
keep-alive: timeout=60
transfer-encoding: chunked


学生控制器 Student Controller
GET
/student/{pageNum}/{pageSize} 分页查询学生的信息
分页查询学生的信息
Parameters
Try it out
Name
Description
pageNum * required
string
(path)
页码
pageSize * required
string
(path)
每页能显示的条数
Responses
Response content type
/
Code
Description

knife4j介绍

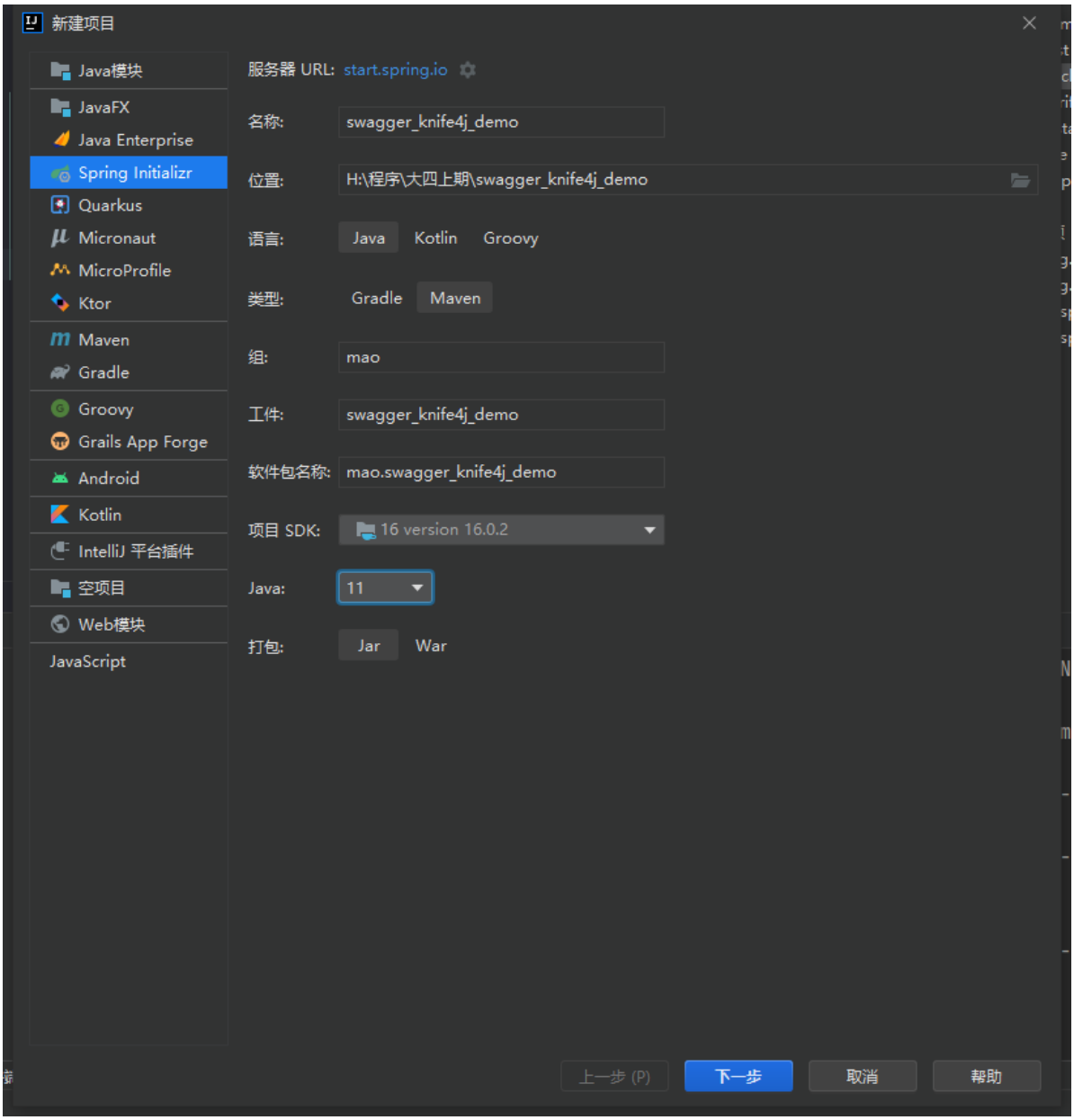
knife4j是为Java MVC框架集成Swagger生成Api文档的增强解决方案,前身是swagger-bootstrap-ui,取名knife4j是希望它能像一把匕首一样小巧,轻量,并且功能强悍!其底层是对Springfox的封装，使用方式也和Springfox一致，只是对接口文档UI进行了优化。

核心功能：

- **文档说明**：根据Swagger的规范说明，详细列出接口文档的说明，包括接口地址、类型、请求示例、请求参数、响应示例、响应参数、响应码等信息，对该接口的使用情况一目了然。
- **在线调试**：提供在线接口联调的强大功能，自动解析当前接口参数,同时包含表单验证，调用参数可返回接口响应内容、headers、响应时间、响应状态码等信息，帮助开发者在线调试。

knife4j入门案例

第一步：创建工程swagger_knife4j_demo

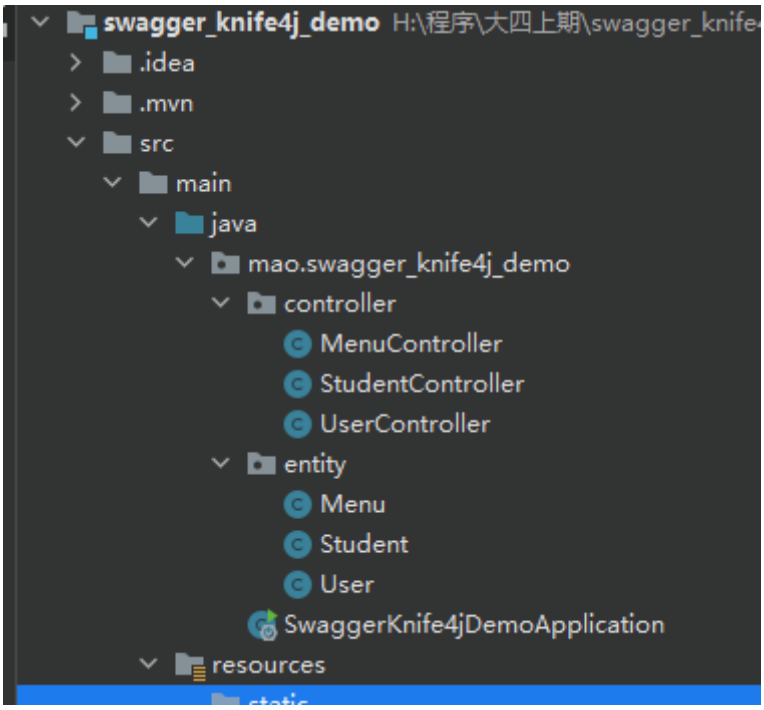


第二步：修改pom文件

```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>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
<!--        <version>2.2.12.RELEASE</version>-->
        <version>2.7.1</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>mao</groupId>
    <artifactId>swagger_knife4j_demo</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>swagger_knife4j_demo</name>
    <description>swagger_knife4j_demo</description>
    <properties>
        <java.version>11</java.version>
    </properties>
    <dependencies>
        <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>
        </dependency>

        <dependency>
            <groupId>com.github.xiaoymin</groupId>
            <artifactId>knife4j-spring-boot-starter</artifactId>
            <version>2.0.1</version>
        </dependency>

    </dependencies>

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

```
48
49 </project>
50
```

第三步：拷贝之前的entity包和controller包到此项目里



第四步：创建配置属性类SwaggerConfigurationProperties

```java
package mao.swagger_knife4j_demo.config;


import org.springframework.boot.context.properties.ConfigurationProperties;
import org.springframework.stereotype.Component;

import java.util.ArrayList;
import java.util.LinkedHashMap;
import java.util.List;
import java.util.Map;

/**
 * Project name(项目名称)：swagger_knife4j_demo
 * Package(包名): mao.swagger_knife4j_demo.config
```

```
 * Class(类名): SwaggerConfigurationProperties
 * Author(作者）: mao
 * Author QQ: 1296193245
 * GitHub: https://github.com/maomao124/
 * Date(创建日期)： 2022/10/27
 * Time(创建时间)： 13:16
 * Version(版本): 1.0
 * Description(描述)： 无
 */

@Component("swaggerConfigurationProperties")
@ConfigurationProperties(prefix = "swagger")
public class SwaggerConfigurationProperties
{
    /**
     * 标题
     */
    private String title = "在线文档";

    /**
     * 自定义组名
     */
    private String group = "";

    /**
     * 描述
     */
    private String description = "在线文档";

    /**
     * 版本
     */
    private String version = "1.0";

    /**
     * 联系人
     */
    private Contact contact = new Contact();

    /**
     * swagger会解析的包路径
     */
    private String basePackage = "";

    /**
     * swagger会解析的url规则
     */
    private List<String> basePath = new ArrayList<>();

    /**
     * 在basePath基础上需要排除的url规则
     */
    private List<String> excludePath = new ArrayList<>();

    /**
     * 分组文档
     */
    private Map<String, DocketInfo> docket = new LinkedHashMap<>();
```

```
73
74
75      public SwaggerConfigurationProperties()
76      {
77
78      }
79
80      /**
81       * 构造方法
82       *
83       * @param title       标题
84       * @param group       组
85       * @param description 描述
86       * @param version     版本
87       * @param contact     联系
88       * @param basePackage 基本包
89       * @param basePath    基本路径
90       * @param excludePath 排除路径
91       * @param docket      摘要
92       */
93      public SwaggerConfigurationProperties(String title, String group,
String description,
94                                            String version, Contact contact,
String basePackage,
95                                            List<String> basePath,
List<String> excludePath,
96                                            Map<String, DocketInfo> docket)
97      {
98          this.title = title;
99          this.group = group;
100         this.description = description;
101         this.version = version;
102         this.contact = contact;
103         this.basePackage = basePackage;
104         this.basePath = basePath;
105         this.excludePath = excludePath;
106         this.docket = docket;
107     }
108
109     /**
110      * 获得组
111      *
112      * @return {@link String}
113      */
114     public String getGroup()
115     {
116         if (group == null || "".equals(group))
117         {
118             return title;
119         }
120         return group;
121     }
122
123
124     /**
125      * 获得标题
126      *
127      * @return {@link String}
```

```
128          */
129         public String getTitle()
130         {
131             return title;
132         }
133 
134         /**
135          * 设置标题
136          *
137          * @param title 标题
138          */
139         public void setTitle(String title)
140         {
141             this.title = title;
142         }
143 
144         /**
145          * 设置组
146          *
147          * @param group 组
148          */
149         public void setGroup(String group)
150         {
151             this.group = group;
152         }
153 
154         /**
155          * 得到描述
156          *
157          * @return {@link String}
158          */
159         public String getDescription()
160         {
161             return description;
162         }
163 
164         /**
165          * 设置描述
166          *
167          * @param description 描述
168          */
169         public void setDescription(String description)
170         {
171             this.description = description;
172         }
173 
174         /**
175          * 获得版本
176          *
177          * @return {@link String}
178          */
179         public String getVersion()
180         {
181             return version;
182         }
183 
184         /**
185          * 设置版本
```

```
186         *
187         * @param version 版本
188         */
189        public void setVersion(String version)
190        {
191            this.version = version;
192        }
193
194        /**
195         * 得到联系
196         *
197         * @return {@link Contact}
198         */
199        public Contact getContact()
200        {
201            return contact;
202        }
203
204        /**
205         * 建立联系
206         *
207         * @param contact 联系
208         */
209        public void setContact(Contact contact)
210        {
211            this.contact = contact;
212        }
213
214        /**
215         * 获得基础包
216         *
217         * @return {@link String}
218         */
219        public String getBasePackage()
220        {
221            return basePackage;
222        }
223
224        /**
225         * 设置基础包
226         *
227         * @param basePackage 基本包
228         */
229        public void setBasePackage(String basePackage)
230        {
231            this.basePackage = basePackage;
232        }
233
234        /**
235         * 得到基本路径
236         *
237         * @return {@link List}<{@link String}>
238         */
239        public List<String> getBasePath()
240        {
241            return basePath;
242        }
243
```

```
    /**
     * 设置基本路径
     *
     * @param basePath 基本路径
     */
    public void setBasePath(List<String> basePath)
    {
        this.basePath = basePath;
    }

    /**
     * 得到排除路径
     *
     * @return {@link List}<{@link String}>
     */
    public List<String> getExcludePath()
    {
        return excludePath;
    }

    /**
     * 设置排除路径
     *
     * @param excludePath 排除路径
     */
    public void setExcludePath(List<String> excludePath)
    {
        this.excludePath = excludePath;
    }

    /**
     * 得到摘要
     *
     * @return {@link Map}<{@link String}, {@link DocketInfo}>
     */
    public Map<String, DocketInfo> getDocket()
    {
        return docket;
    }

    /**
     * 设置摘要
     *
     * @param docket 摘要
     */
    public void setDocket(Map<String, DocketInfo> docket)
    {
        this.docket = docket;
    }

    public static class DocketInfo
    {
        /**
         * 标题
         */
        private String title = "在线文档";

        /**
```

```
302             * 自定义组名
303             */
304            private String group = "";
305
306            /**
307             * 描述
308             */
309            private String description = "在线文档";
310
311            /**
312             * 版本
313             */
314            private String version = "1.0";
315
316            /**
317             * 联系人
318             */
319            private Contact contact = new Contact();
320
321            /**
322             * swagger会解析的包路径
323             */
324            private String basePackage = "";
325
326            /**
327             * swagger会解析的url规则
328             */
329            private List<String> basePath = new ArrayList<>();
330
331            /**
332             * 在basePath基础上需要排除的url
333             */
334            private List<String> excludePath = new ArrayList<>();
335
336
337            public DocketInfo()
338            {
339
340            }
341
342            /**
343             * 构造方法
344             *
345             * @param title       标题
346             * @param group       组
347             * @param description 描述
348             * @param version     版本
349             * @param contact     联系
350             * @param basePackage 基本包
351             * @param basePath    基本路径
352             * @param excludePath 排除路径
353             */
354            public DocketInfo(String title, String group, String description,
355                              String version, Contact contact, String
    basePackage,
356                              List<String> basePath, List<String> excludePath)
357            {
358                this.title = title;
```

```
359             this.group = group;
360             this.description = description;
361             this.version = version;
362             this.contact = contact;
363             this.basePackage = basePackage;
364             this.basePath = basePath;
365             this.excludePath = excludePath;
366         }
367 
368         /**
369          * 获得组
370          *
371          * @return {@link String}
372          */
373         public String getGroup()
374         {
375             if (group == null || "".equals(group))
376             {
377                 return title;
378             }
379             return group;
380         }
381 
382         /**
383          * 获得标题
384          *
385          * @return {@link String}
386          */
387         public String getTitle()
388         {
389             return title;
390         }
391 
392         /**
393          * 设置标题
394          *
395          * @param title 标题
396          */
397         public void setTitle(String title)
398         {
399             this.title = title;
400         }
401 
402         /**
403          * 设置组
404          *
405          * @param group 组
406          */
407         public void setGroup(String group)
408         {
409             this.group = group;
410         }
411 
412         /**
413          * 得到描述
414          *
415          * @return {@link String}
416          */
```

```
417         public String getDescription()
418         {
419             return description;
420         }
421 
422         /**
423          * 设置描述
424          *
425          * @param description 描述
426          */
427         public void setDescription(String description)
428         {
429             this.description = description;
430         }
431 
432         /**
433          * 获得版本
434          *
435          * @return {@link String}
436          */
437         public String getVersion()
438         {
439             return version;
440         }
441 
442         /**
443          * 设置版本
444          *
445          * @param version 版本
446          */
447         public void setVersion(String version)
448         {
449             this.version = version;
450         }
451 
452         /**
453          * 得到联系
454          *
455          * @return {@link Contact}
456          */
457         public Contact getContact()
458         {
459             return contact;
460         }
461 
462         /**
463          * 建立联系
464          *
465          * @param contact 联系
466          */
467         public void setContact(Contact contact)
468         {
469             this.contact = contact;
470         }
471 
472         /**
473          * 获得基础包
474          *
```

```
475             * @return {@link String}
476             */
477            public String getBasePackage()
478            {
479                return basePackage;
480            }
481
482            /**
483             * 设置基础包
484             *
485             * @param basePackage 基本包
486             */
487            public void setBasePackage(String basePackage)
488            {
489                this.basePackage = basePackage;
490            }
491
492            /**
493             * 得到基本路径
494             *
495             * @return {@link List}<{@link String}>
496             */
497            public List<String> getBasePath()
498            {
499                return basePath;
500            }
501
502            /**
503             * 设置基本路径
504             *
505             * @param basePath 基本路径
506             */
507            public void setBasePath(List<String> basePath)
508            {
509                this.basePath = basePath;
510            }
511
512            /**
513             * 得到排除路径
514             *
515             * @return {@link List}<{@link String}>
516             */
517            public List<String> getExcludePath()
518            {
519                return excludePath;
520            }
521
522            /**
523             * 设置排除路径
524             *
525             * @param excludePath 排除路径
526             */
527            public void setExcludePath(List<String> excludePath)
528            {
529                this.excludePath = excludePath;
530            }
531        }
532
```

```
533     public static class Contact
534     {
535         /**
536          * 联系人名称
537          */
538         private String name = "";
539 
540         /**
541          * url
542          */
543         private String url = "";
544 
545         /**
546          * 电子邮件
547          */
548         private String email = "";
549 
550 
551         public Contact()
552         {
553 
554         }
555 
556         /**
557          * 构造方法
558          *
559          * @param name  名字
560          * @param url   url
561          * @param email 电子邮件
562          */
563         public Contact(String name, String url, String email)
564         {
565             this.name = name;
566             this.url = url;
567             this.email = email;
568         }
569 
570         /**
571          * 得到名字
572          *
573          * @return {@link String}
574          */
575         public String getName()
576         {
577             return name;
578         }
579 
580         /**
581          * 设置名字
582          *
583          * @param name 名字
584          */
585         public void setName(String name)
586         {
587             this.name = name;
588         }
589 
590         /**
```

```
591             * 获取url
592             *
593             * @return {@link String}
594             */
595            public String getUrl()
596            {
597                return url;
598            }
599
600            /**
601             * 设置url
602             *
603             * @param url url
604             */
605            public void setUrl(String url)
606            {
607                this.url = url;
608            }
609
610            /**
611             * 获得电子邮件
612             *
613             * @return {@link String}
614             */
615            public String getEmail()
616            {
617                return email;
618            }
619
620            /**
621             * 设置电子邮件
622             *
623             * @param email 电子邮件
624             */
625            public void setEmail(String email)
626            {
627                this.email = email;
628            }
629        }
630
631
632        @Override
633        public boolean equals(Object o)
634        {
635            if (this == o)
636            {
637                return true;
638            }
639            if (o == null || getClass() != o.getClass())
640            {
641                return false;
642            }
643
644            SwaggerConfigurationProperties that =
(SwaggerConfigurationProperties) o;
645
646            if (getTitle() != null ? !getTitle().equals(that.getTitle()) :
that.getTitle() != null)
```

```
        {
            return false;
        }
        if (getGroup() != null ? !getGroup().equals(that.getGroup()) :
that.getGroup() != null)
        {
            return false;
        }
        if (getDescription() != null ?
!getDescription().equals(that.getDescription()) : that.getDescription() !=
null)
        {
            return false;
        }
        if (getVersion() != null ? !getVersion().equals(that.getVersion())
: that.getVersion() != null)
        {
            return false;
        }
        if (getContact() != null ? !getContact().equals(that.getContact())
: that.getContact() != null)
        {
            return false;
        }
        if (getBasePackage() != null ?
!getBasePackage().equals(that.getBasePackage()) : that.getBasePackage() !=
null)
        {
            return false;
        }
        if (getBasePath() != null ?
!getBasePath().equals(that.getBasePath()) : that.getBasePath() != null)
        {
            return false;
        }
        if (getExcludePath() != null ?
!getExcludePath().equals(that.getExcludePath()) : that.getExcludePath() !=
null)
        {
            return false;
        }
        return getDocket() != null ? getDocket().equals(that.getDocket()) :
that.getDocket() == null;
    }

    @Override
    public int hashCode()
    {
        int result = getTitle() != null ? getTitle().hashCode() : 0;
        result = 31 * result + (getGroup() != null ? getGroup().hashCode()
: 0);
        result = 31 * result + (getDescription() != null ?
getDescription().hashCode() : 0);
        result = 31 * result + (getVersion() != null ?
getVersion().hashCode() : 0);
        result = 31 * result + (getContact() != null ?
getContact().hashCode() : 0);
```

```
689         result = 31 * result + (getBasePackage() != null ?
    getBasePackage().hashCode() : 0);
690         result = 31 * result + (getBasePath() != null ?
    getBasePath().hashCode() : 0);
691         result = 31 * result + (getExcludePath() != null ?
    getExcludePath().hashCode() : 0);
692         result = 31 * result + (getDocket() != null ?
    getDocket().hashCode() : 0);
693         return result;
694     }
695
696     @Override
697     @SuppressWarnings("all")
698     public String toString()
699     {
700         final StringBuilder stringbuilder = new StringBuilder();
701         stringbuilder.append("title: ").append(title).append('\n');
702         stringbuilder.append("group: ").append(group).append('\n');
703
     stringbuilder.append("description: ").append(description).append('\n');
704         stringbuilder.append("version: ").append(version).append('\n');
705         stringbuilder.append("contact: ").append(contact).append('\n');
706
     stringbuilder.append("basePackage: ").append(basePackage).append('\n');
707         stringbuilder.append("basePath: ").append(basePath).append('\n');
708
     stringbuilder.append("excludePath: ").append(excludePath).append('\n');
709         stringbuilder.append("docket: ").append(docket).append('\n');
710         return stringbuilder.toString();
711     }
712 }
```

第五步：修改application.yml文件

```
1  swagger:
2    # 是否启用swagger
3    enabled: true
4    title: 在线文档
5    group: 默认组
6    version: 1.0
7    contact:
8      name: mao
9      url: https://github.com/maomao124/
10     email: 1234@qq.com
11   base-package: mao.swagger_knife4j_demo
12   # 分组文档
13 #  docket:
14 #    user:
```

```
#      title: 用户模块
#      base-package:
#    menu:
#      title: 菜单模块
#      base-package:


spring:
  mvc:
    pathmatch:
      matching-strategy: ant_path_matcher
```

第六步：创建配置类SwaggerConfig

```
package mao.swagger_knife4j_demo.config;

import com.google.common.base.Predicate;
import com.google.common.base.Predicates;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.BeansException;
import org.springframework.beans.factory.BeanFactory;
import org.springframework.beans.factory.BeanFactoryAware;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.config.ConfigurableBeanFactory;
import
org.springframework.boot.autoconfigure.condition.ConditionalOnMissingBean;
import
org.springframework.boot.autoconfigure.condition.ConditionalOnProperty;
import
org.springframework.boot.context.properties.EnableConfigurationProperties;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.Import;
import springfox.documentation.builders.ApiInfoBuilder;
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 javax.annotation.PostConstruct;
import java.util.ArrayList;
import java.util.LinkedList;
import java.util.List;

/**
```

```
 * Project name(项目名称)：swagger_knife4j_demo
 * Package(包名): mao.swagger_knife4j_demo.config
 * Class(类名): SwaggerConfig
 * Author(作者）: mao
 * Author QQ：1296193245
 * GitHub：https://github.com/maomao124/
 * Date(创建日期)： 2022/10/27
 * Time(创建时间)： 13:40
 * Version(版本): 1.0
 * Description(描述)： 无
 */

@Configuration
@EnableSwagger2
@ConditionalOnProperty(name = "swagger.enabled", havingValue = "true",
matchIfMissing = true)
@EnableConfigurationProperties
@Import(SwaggerConfigurationProperties.class)
public class SwaggerConfig implements BeanFactoryAware
{

    /*

    配置文件示例：

swagger:
  # 是否启用swagger
  enabled: true
  title: 在线文档
  group: 默认组
  version: 1.0
  contact:
    name: mao
    url: https://github.com/maomao124/
    email: 1234@qq.com
  base-package: mao.swagger_knife4j_demo
  # 分组文档
#  docket:
#    user:
#      title: 用户模块
#      base-package:
#    menu:
#      title: 菜单模块
#      base-package:



spring:
  mvc:
    pathmatch:
      matching-strategy: ant_path_matcher



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

```

```
90
91      @Autowired
92      private SwaggerConfigurationProperties swaggerConfigurationProperties;
93
94      private static final Logger log =
LoggerFactory.getLogger(SwaggerConfig.class);
95
96      /**
97       * bean工厂
98       */
99      private BeanFactory beanFactory;
100
101     @Override
102     public void setBeanFactory(BeanFactory beanFactory) throws
BeansException
103     {
104         this.beanFactory = beanFactory;
105     }
106
107
108     @Bean
109     @ConditionalOnMissingBean
110     @SuppressWarnings("all")
111     public List<Docket> createRestApi()
112     {
113         ConfigurableBeanFactory configurableBeanFactory =
114                 (ConfigurableBeanFactory) beanFactory;
115         List<Docket> docketList = new LinkedList<>();
116         // 没有分组的情况
117         if (swaggerConfigurationProperties.getDocket().isEmpty())
118         {
119             Docket docket = createDocket(swaggerConfigurationProperties);
120
 configurableBeanFactory.registerSingleton(swaggerConfigurationProperties.g
etTitle(),
121                     docket);
122             docketList.add(docket);
123             return docketList;
124         }
125         // 分组创建
126         for (String groupName :
swaggerConfigurationProperties.getDocket().keySet())
127         {
128             SwaggerConfigurationProperties.DocketInfo docketInfo =
129
 swaggerConfigurationProperties.getDocket().get(groupName);
130             ApiInfo apiInfo = new ApiInfoBuilder()
131                     //页面标题
132                     .title(docketInfo.getTitle())
133                     //创建人
134                     .contact(new Contact
135                             (
136                                     docketInfo.getContact().getName(),
137                                     docketInfo.getContact().getUrl(),
138                                     docketInfo.getContact().getEmail()))
139                     //版本号
140                     .version(docketInfo.getVersion())
141                     //描述
```

```
                    .description(docketInfo.getDescription())
                    .build();

            // base-path处理
            // 当没有配置任何path的时候，解析/**
            if (docketInfo.getBasePath().isEmpty())
            {
                docketInfo.getBasePath().add("/**");
            }
            List<Predicate<String>> basePath = new ArrayList<>();
            for (String path : docketInfo.getBasePath())
            {
                basePath.add(PathSelectors.ant(path));
            }

            // exclude-path处理
            List<Predicate<String>> excludePath = new ArrayList<>();
            for (String path : docketInfo.getExcludePath())
            {
                excludePath.add(PathSelectors.ant(path));
            }

            Docket docket = new Docket(DocumentationType.SWAGGER_2)
                    .apiInfo(apiInfo)
                    .groupName(docketInfo.getGroup())
                    .select()
                    //为当前包路径

.apis(RequestHandlerSelectors.basePackage(docketInfo.getBasePackage()))

.paths(Predicates.and(Predicates.not(Predicates.or(excludePath)),
Predicates.or(basePath)))
                    .build();
            configurableBeanFactory.registerSingleton(groupName, docket);
            docketList.add(docket);
        }
        return docketList;
    }

    /**
     * 构建 api文档的详细信息
     *
     * @param swaggerConfigurationProperties 配置属性
     * @return {@link ApiInfo}
     */
    private ApiInfo apiInfo(SwaggerConfigurationProperties
swaggerConfigurationProperties)
    {
        return new ApiInfoBuilder()
                //页面标题
                .title(swaggerConfigurationProperties.getTitle())
                //创建人
                .contact(new Contact
                        (

 swaggerConfigurationProperties.getContact().getName(),

 swaggerConfigurationProperties.getContact().getUrl(),
```

```
 swaggerConfigurationProperties.getContact().getEmail()
                        )
                )
                //版本号
                .version(swaggerConfigurationProperties.getVersion())
                //描述

.description(swaggerConfigurationProperties.getDescription())
                .build();
    }


    /**
     * 创建接口文档对象
     *
     * @param swaggerConfigurationProperties 配置属性
     * @return {@link Docket}
     */
    @SuppressWarnings("all")
    private Docket createDocket(SwaggerConfigurationProperties
swaggerConfigurationProperties)
    {
        //API 基础信息
        ApiInfo apiInfo = apiInfo(swaggerConfigurationProperties);

        // base-path处理
        // 当没有配置任何path的时候，解析/**
        if (swaggerConfigurationProperties.getBasePath().isEmpty())
        {
            swaggerConfigurationProperties.getBasePath().add("/**");
        }
        List<Predicate<String>> basePath = new ArrayList<>();
        for (String path : swaggerConfigurationProperties.getBasePath())
        {
            basePath.add(PathSelectors.ant(path));
        }

        // exclude-path处理
        List<Predicate<String>> excludePath = new ArrayList<>();
        for (String path : swaggerConfigurationProperties.getExcludePath())
        {
            excludePath.add(PathSelectors.ant(path));
        }

        return new Docket(DocumentationType.SWAGGER_2)
                .apiInfo(apiInfo)
                .groupName(swaggerConfigurationProperties.getGroup())
                .select()

.apis(RequestHandlerSelectors.basePackage(swaggerConfigurationProperties.ge
tBasePackage()))

.paths(Predicates.and(Predicates.not(Predicates.or(excludePath)),
Predicates.or(basePath)))
                .build();
    }

```

```
245
246     @PostConstruct
247     public void init()
248     {
249         log.info("初始化swagger接口文档");
250     }
251
252 }
```

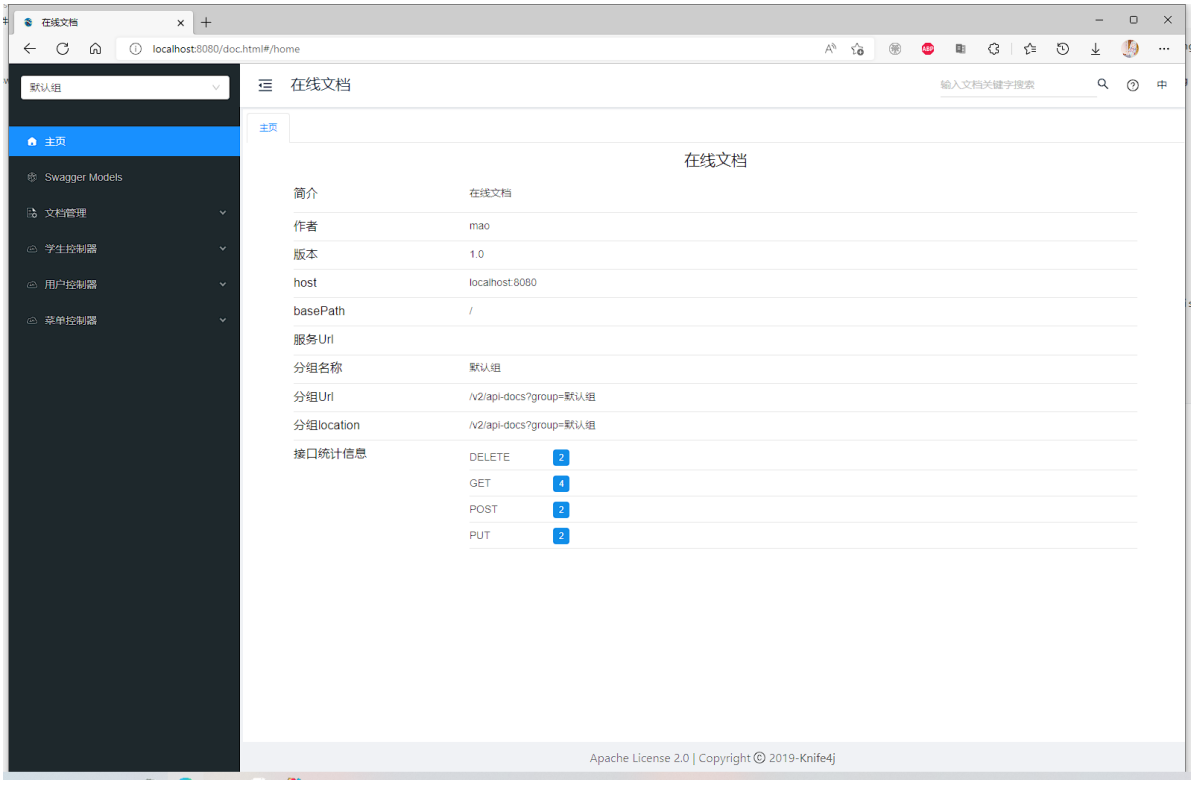
第七步：启动程序

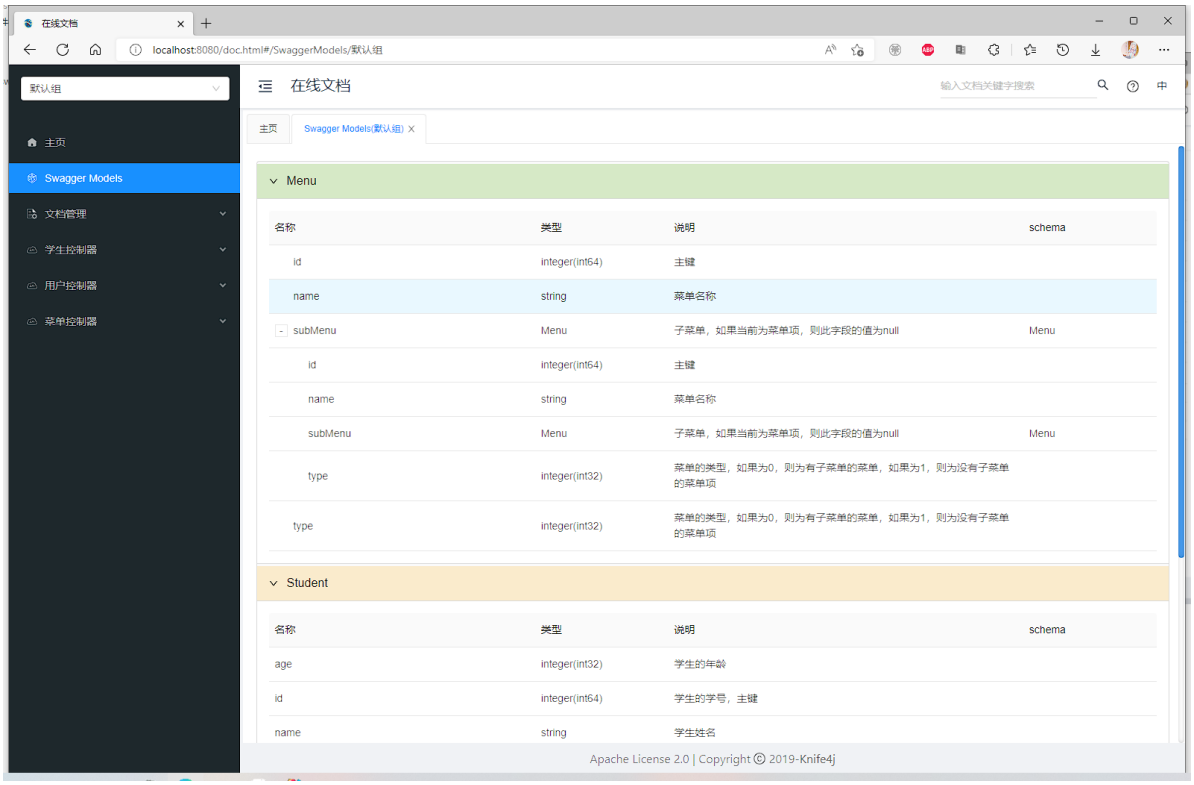
```
1
2   .   ____          _            __ _ _
3  /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
4 ( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
5  \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
6   '  |____| .__|_| |_|_| |_\__, | / / / /
7  =========|_|==============|___/=/_/_/_/
8  :: Spring Boot ::                (v2.7.1)
9
10 2022-10-27 14:44:17.993  INFO 11532 --- [           main]
   m.s.SwaggerKnife4jDemoApplication        : Starting
   SwaggerKnife4jDemoApplication using Java 16.0.2 on mao with PID 11532 (H:\程
   序\大四上期\swagger_knife4j_demo\target\classes started by mao in H:\程序\大四
   上期\swagger_knife4j_demo)
11 2022-10-27 14:44:17.996  INFO 11532 --- [           main]
   m.s.SwaggerKnife4jDemoApplication        : No active profile set, falling
   back to 1 default profile: "default"
12 2022-10-27 14:44:18.791  INFO 11532 --- [           main]
   o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s):
   8080 (http)
13 2022-10-27 14:44:18.797  INFO 11532 --- [           main]
   o.apache.catalina.core.StandardService   : Starting service [Tomcat]
14 2022-10-27 14:44:18.797  INFO 11532 --- [           main]
   org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Apache
   Tomcat/9.0.64]
15 2022-10-27 14:44:18.882  INFO 11532 --- [           main] o.a.c.c.C.
   [Tomcat].[localhost].[/]       : Initializing Spring embedded
   WebApplicationContext
16 2022-10-27 14:44:18.882  INFO 11532 --- [           main]
   w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext:
   initialization completed in 846 ms
17 2022-10-27 14:44:18.927  INFO 11532 --- [           main]
   m.s.config.SwaggerConfig                 : 初始化swagger接口文档
18 2022-10-27 14:44:19.135  INFO 11532 --- [           main]
   pertySourcedRequestMappingHandlerMapping : Mapped URL path [/v2/api-docs]
   onto method
   [springfox.documentation.swagger2.web.Swagger2Controller#getDocumentation(St
   ring, HttpServletRequest)]
```

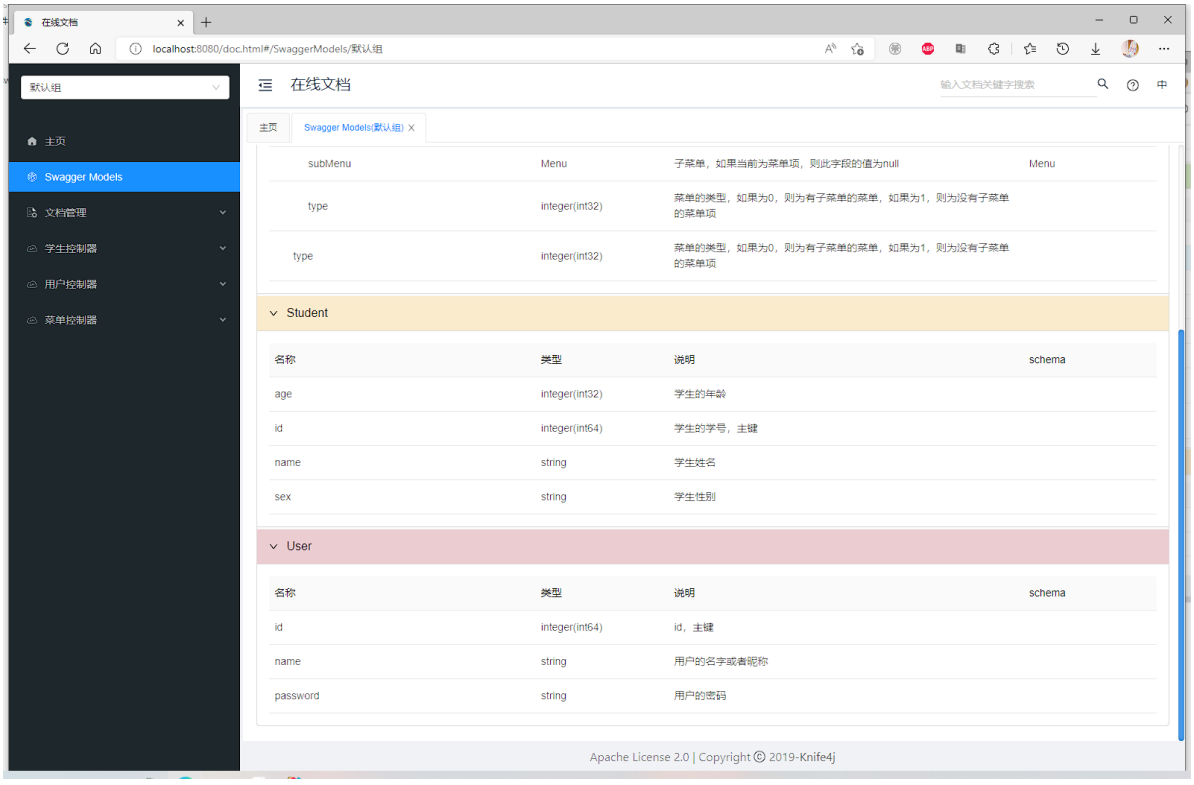
```
2022-10-27 14:44:19.289  INFO 11532 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8080 (http) with context path ''
2022-10-27 14:44:19.290  INFO 11532 --- [           main] d.s.w.p.DocumentationPluginsBootstrapper : Context refreshed
2022-10-27 14:44:19.300  INFO 11532 --- [           main] d.s.w.p.DocumentationPluginsBootstrapper : Found 1 custom documentation plugin(s)
2022-10-27 14:44:19.320  INFO 11532 --- [           main] s.d.s.w.s.ApiListingReferenceScanner     : Scanning for api listing references
2022-10-27 14:44:19.414  INFO 11532 --- [           main] .d.s.w.r.o.CachingOperationNameGenerator : Generating unique operation named: findByPageUsingGET_1
2022-10-27 14:44:19.443  INFO 11532 --- [           main] m.s.SwaggerKnife4jDemoApplication        : Started SwaggerKnife4jDemoApplication in 1.749 seconds (JVM running for 2.216)
```

第八步：访问

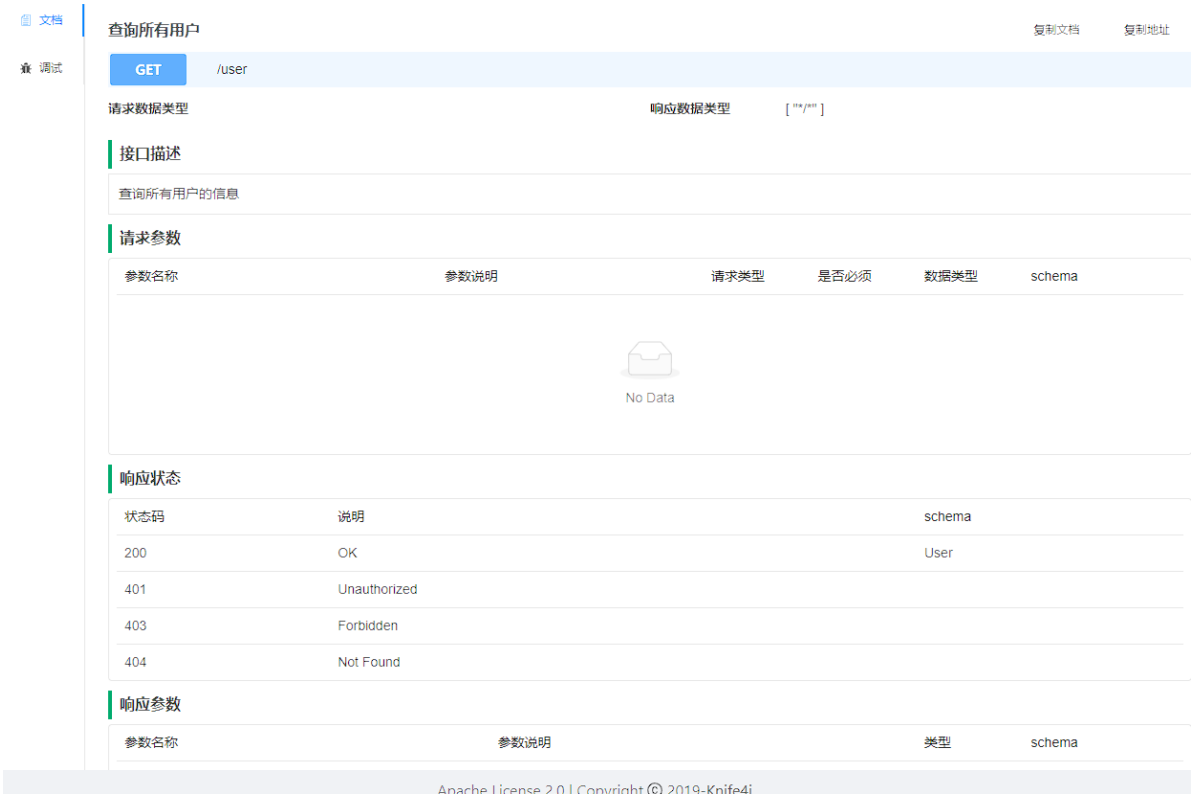
http://localhost:8080/doc.html




在线文档
localhost:8080/doc.html#/SwaggerModels/默认组
默认组
在线文档
输入文档关键字搜索
主页
Swagger Models
文档管理
学生控制器
用户控制器
菜单控制器
主页
Swagger Models(默认组)
Menu
名称 类型 说明 schema
id integer(int64) 主键
name string 菜单名称
subMenu Menu 子菜单，如果当前为菜单项，则此字段的值为null Menu
id integer(int64) 主键
name string 菜单名称
subMenu Menu 子菜单，如果当前为菜单项，则此字段的值为null Menu
type integer(int32) 菜单的类型，如果为0，则为有子菜单的菜单，如果为1，则为没有子菜单的菜单项
type integer(int32) 菜单的类型，如果为0，则为有子菜单的菜单，如果为1，则为没有子菜单的菜单项
Student
名称 类型 说明 schema
age integer(int32) 学生的年龄
id integer(int64) 学生的学号，主键
name string 学生姓名
Apache License 2.0 | Copyright © 2019-Knife4j


在线文档
localhost:8080/doc.html#/SwaggerModels/默认组
默认组
在线文档
输入文档关键字搜索
主页
Swagger Models
文档管理
学生控制器
用户控制器
菜单控制器
主页
Swagger Models(默认组)
subMenu Menu 子菜单，如果当前为菜单项，则此字段的值为null Menu
type integer(int32) 菜单的类型，如果为0，则为有子菜单的菜单，如果为1，则为没有子菜单的菜单项
type integer(int32) 菜单的类型，如果为0，则为有子菜单的菜单，如果为1，则为没有子菜单的菜单项
Student
名称 类型 说明 schema
age integer(int32) 学生的年龄
id integer(int64) 学生的学号，主键
name string 学生姓名
sex string 学生性别
User
名称 类型 说明 schema
id integer(int64) id，主键
name string 用户的名字或者昵称
password string 用户的密码
Apache License 2.0 | Copyright © 2019-Knife4j


默认组
主页
Swagger Models
文档管理
学生控制器
用户控制器
查询所有用户
保存(添加)用户的信息
更新用户信息
删除用户的信息
根据用户id查询用户
菜单控制器



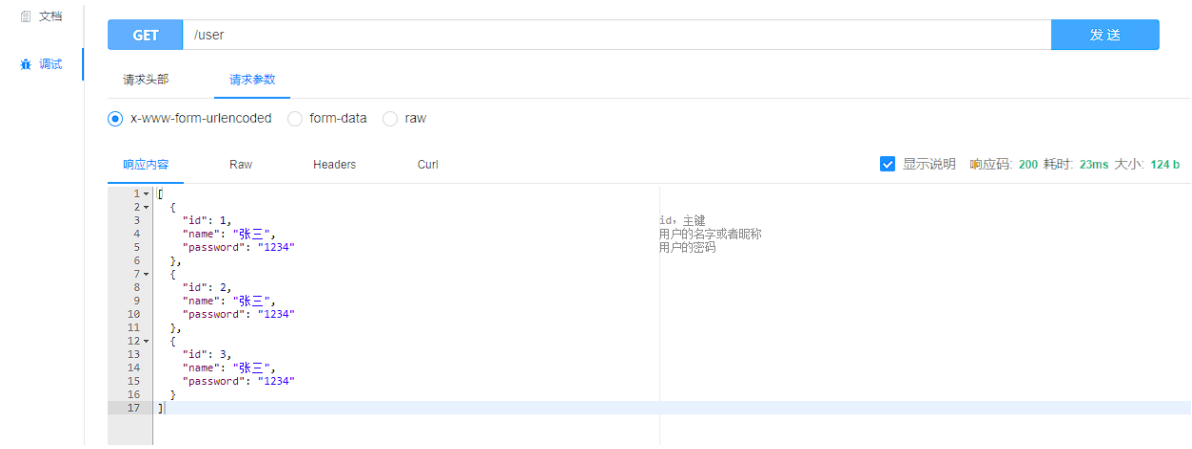
响应参数

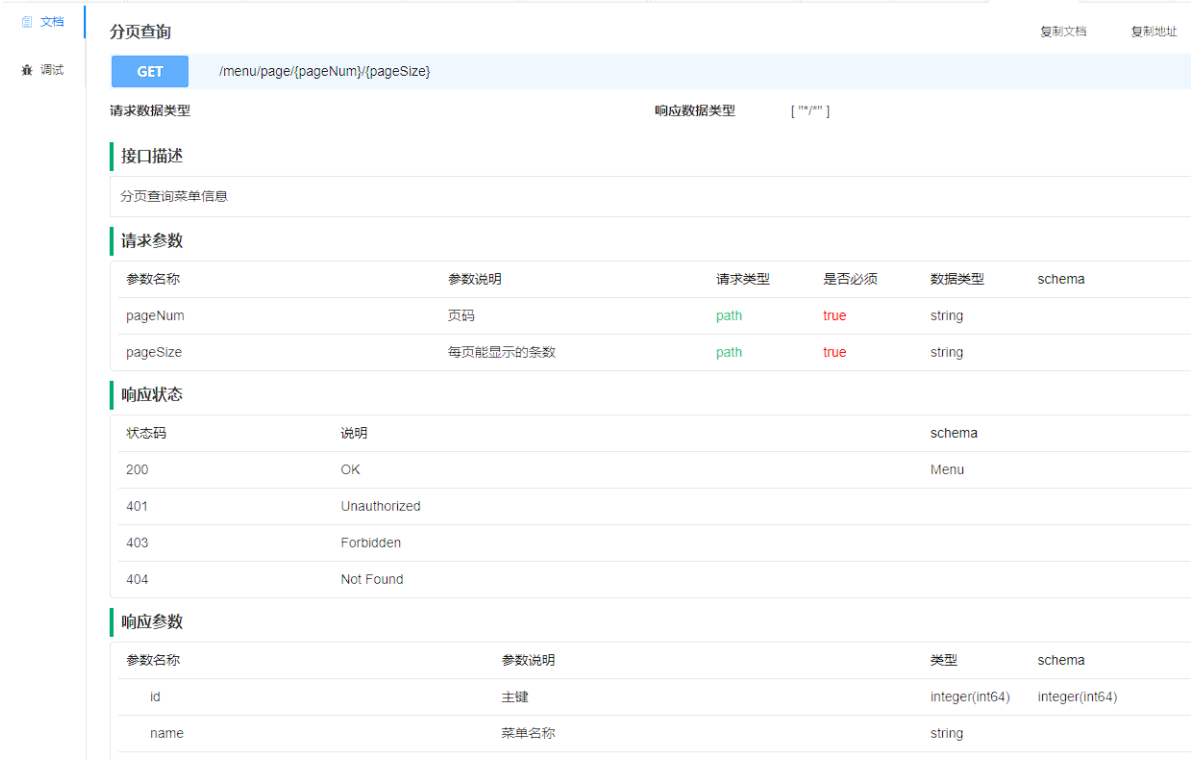
参数名称	参数说明	类型	schema
id	id，主键	integer(int64)	integer(int64)
name	用户的名字或者昵称	string	
password	用户的密码	string	

响应示例

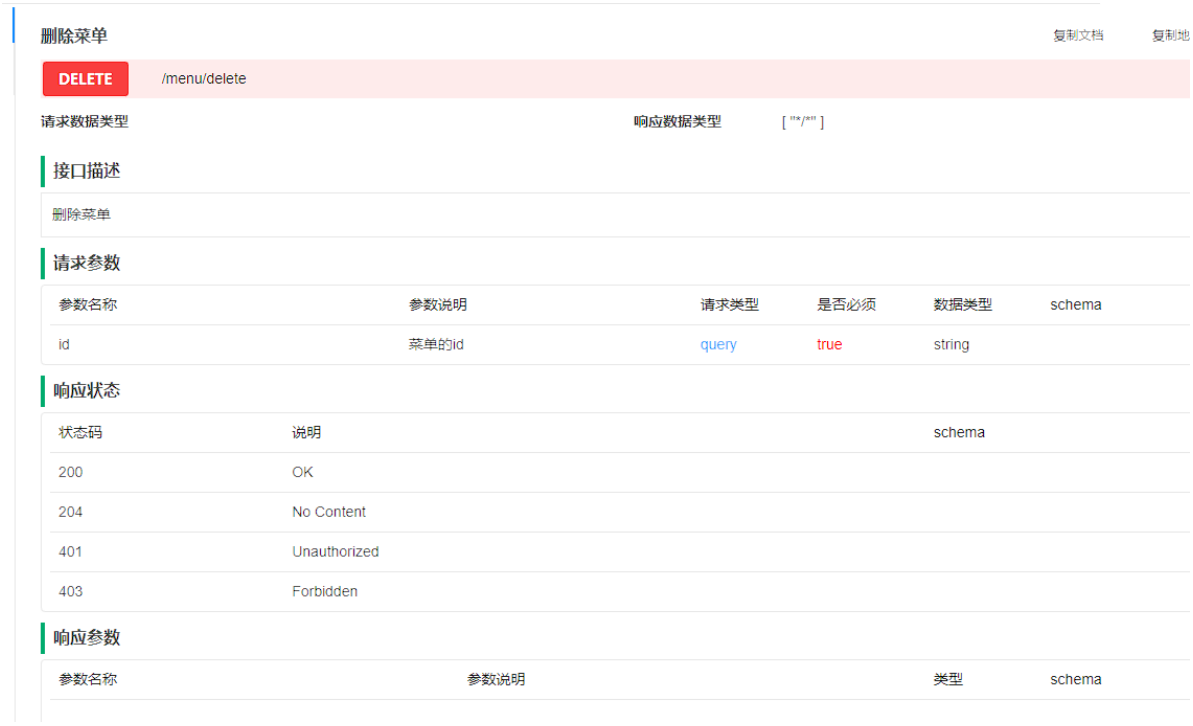
```
[
    {
        "id": 0,
        "name": "",
        "password": ""
    }
]
```

id，主键
用户的名字或者昵称
用户的密码




文档
调试
分页查询
复制文档
复制地址
GET
/menu/page/{pageNum}/{pageSize}
请求数据类型
响应数据类型
["*/*"]
接口描述
分页查询菜单信息
请求参数
参数名称 参数说明 请求类型 是否必须 数据类型 schema
pageNum 页码 path true string
pageSize 每页能显示的条数 path true string
响应状态
状态码 说明 schema
200 OK Menu
401 Unauthorized
403 Forbidden
404 Not Found
响应参数
参数名称 参数说明 类型 schema
id 主键 integer(int64) integer(int64)
name 菜单名称 string


文档
调试
GET
/menu/page/{pageNum}/{pageSize}
发送
请求头部
请求参数
x-www-form-urlencoded
form-data
raw
参数名称
参数值
操作
pageNum
参数值
删除
pageSize
参数值
删除



删除菜单

复制文档 复制地

DELETE /menu/delete

请求数据类型 响应数据类型 ["*/*"]

接口描述

删除菜单

请求参数

参数名称	参数说明	请求类型	是否必须	数据类型	schema
id	菜单的id	query	true	string	

响应状态

状态码	说明	schema
200	OK	
204	No Content	
401	Unauthorized	
403	Forbidden	

响应参数

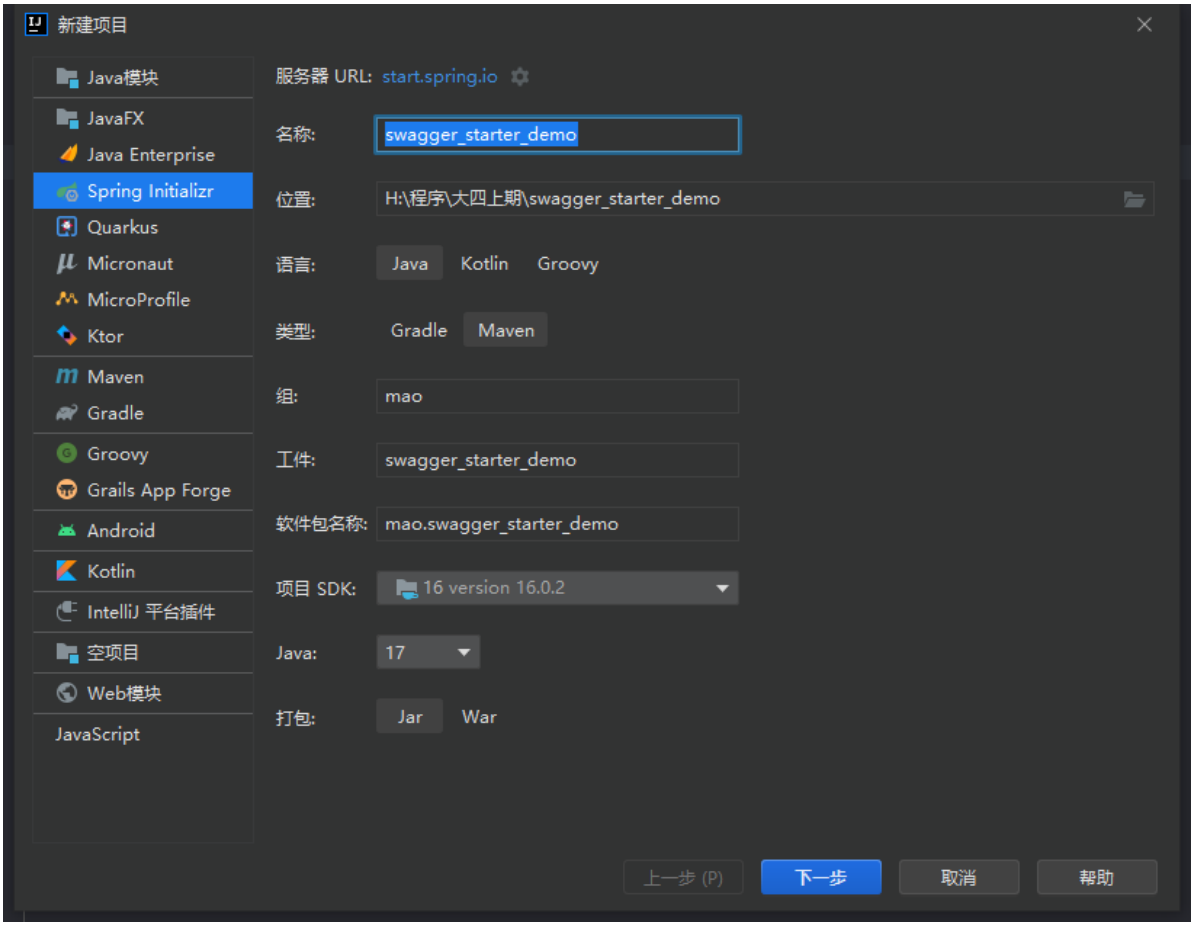
参数名称	参数说明	类型	schema

自定义spring boot starter

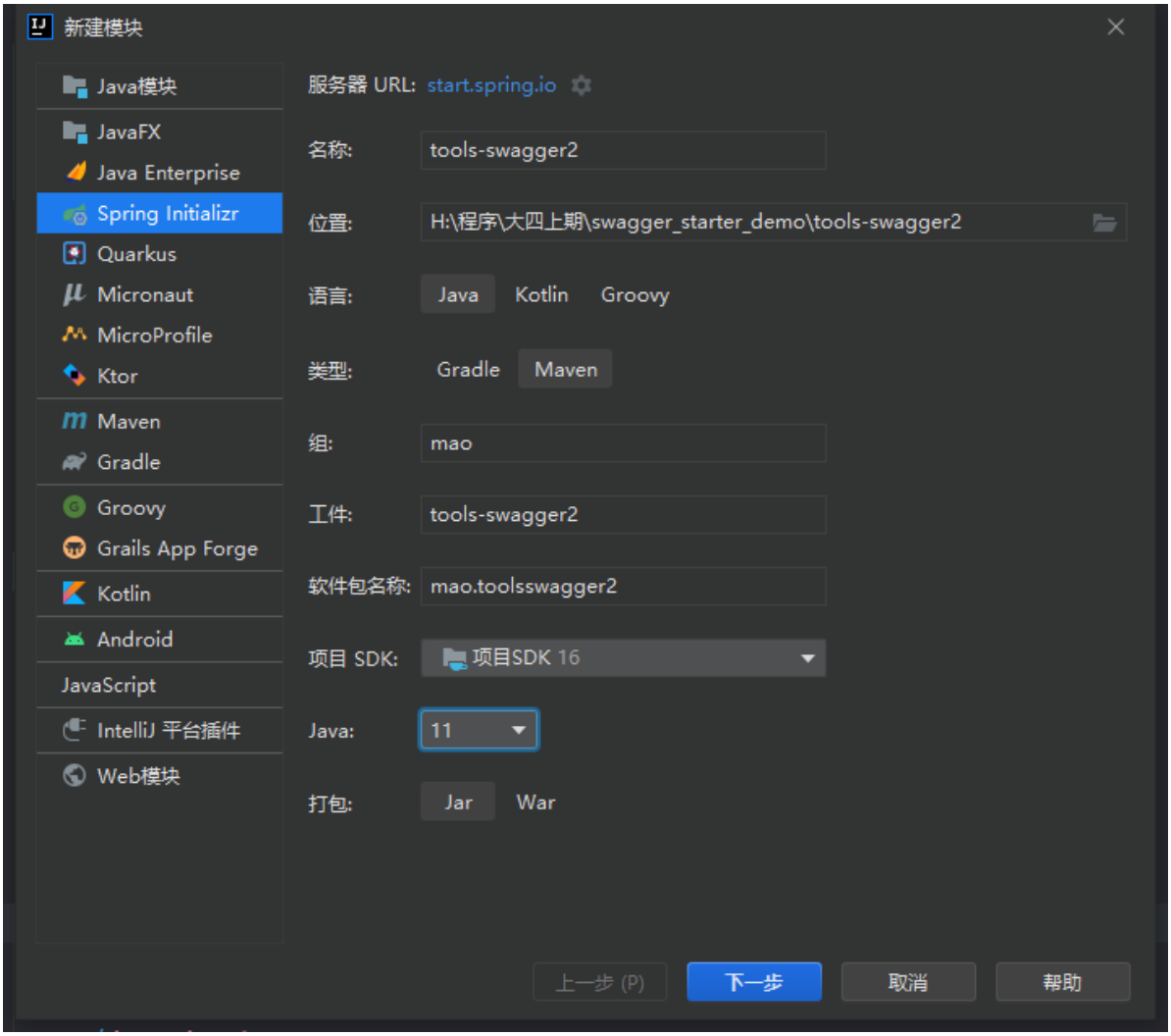
开发starter

第一步：初始化项目

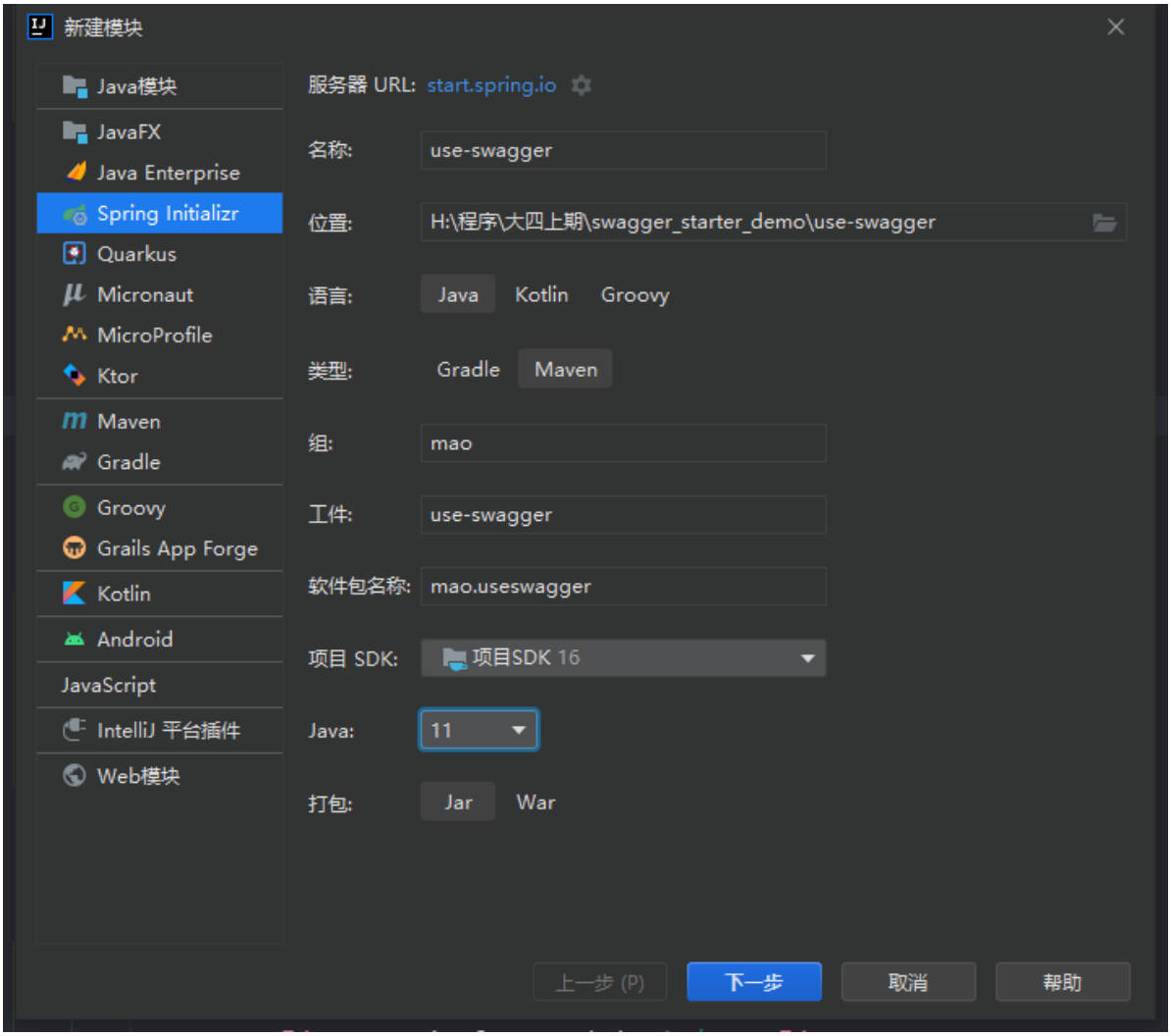
创建父工程swagger_starter_demo



创建子工程tools-swagger2



创建子工程use-swagger



第二步：修改pom文件

父工程swagger_starter_demo：

```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>

    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.7.1</version>
        <relativePath/>
    </parent>

    <groupId>mao</groupId>
    <artifactId>swagger_starter_demo</artifactId>
    <version>0.0.1</version>
```

```
    <name>swagger_starter_demo</name>
    <description>swagger_starter_demo</description>
    <packaging>pom</packaging>

    <properties>
        <java.version>11</java.version>
    </properties>


    <modules>
        <module>tools-swagger2</module>
        <module>use-swagger</module>
    </modules>

    <dependencies>


    </dependencies>

    <dependencyManagement>

        <dependencies>

        </dependencies>

    </dependencyManagement>

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

</project>
```

子工程tools-swagger2：

```
<?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>
        <artifactId>swagger_starter_demo</artifactId>
        <groupId>mao</groupId>
        <version>0.0.1</version>
    </parent>

    <artifactId>tools-swagger2</artifactId>
    <version>0.0.1</version>
```

```
    <name>tools-swagger2</name>
    <description>tools-swagger2</description>

    <properties>

    </properties>

    <dependencies>

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

    </dependencies>

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

</project>
```

子工程use-swagger：

```
<?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>
        <artifactId>swagger_starter_demo</artifactId>
        <groupId>mao</groupId>
        <version>0.0.1</version>
    </parent>

    <artifactId>use-swagger</artifactId>
    <version>0.0.1</version>
    <name>use-swagger</name>
    <description>use-swagger</description>

    <properties>

    </properties>

```

```
    <dependencies>

        <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>
        </dependency>

    </dependencies>

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

</project>
```

第三步：给子工程tools-swagger2添加依赖：

```
        <dependency>
            <groupId>com.github.xiaoymin</groupId>
            <artifactId>knife4j-spring-boot-starter</artifactId>
            <version>2.0.1</version>
        </dependency>

        <!--spring boot starter开发依赖-->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter</artifactId>
        </dependency>

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

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-configuration-processor</artifactId>
        </dependency>
```

第四步：创建配置属性类SwaggerConfigurationProperties

就是之前写的那个类，不重复添加了

第五步：创建配置类SwaggerAutoConfiguration

就是之前写的那个类，内容相同，但是类名不同

```
package mao.toolsswagger2.config;

import com.google.common.base.Predicate;
import com.google.common.base.Predicates;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.BeansException;
import org.springframework.beans.factory.BeanFactory;
import org.springframework.beans.factory.BeanFactoryAware;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.config.ConfigurableBeanFactory;
import
org.springframework.boot.autoconfigure.condition.ConditionalOnMissingBean;
import
org.springframework.boot.autoconfigure.condition.ConditionalOnProperty;
import
org.springframework.boot.context.properties.EnableConfigurationProperties;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.Import;
import springfox.documentation.builders.ApiInfoBuilder;
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 javax.annotation.PostConstruct;
import java.util.ArrayList;
import java.util.LinkedList;
import java.util.List;

/**
 * Project name(项目名称)：tools-swagger2
 * Package(包名):
```

```
 * Class(类名): SwaggerAutoConfiguration
 * Author(作者）: mao.toolsswagger2.config
 * Author QQ：1296193245
 * GitHub：https://github.com/maomao124/
 * Date(创建日期)： 2022/10/27
 * Time(创建时间)： 13:40
 * Version(版本): 1.0
 * Description(描述)： 无
 */

@Configuration
@EnableSwagger2
@ConditionalOnProperty(name = "swagger.enabled", havingValue = "true",
matchIfMissing = true)
@EnableConfigurationProperties
@Import(SwaggerConfigurationProperties.class)
public class SwaggerAutoConfiguration implements BeanFactoryAware
{

    /*

    配置文件示例：

swagger:
  # 是否启用swagger
  enabled: true
  title: 在线文档
  group: 默认组
  version: 1.0
  contact:
    name: mao
    url: https://github.com/maomao124/
    email: 1234@qq.com
  base-package: mao.swagger_knife4j_demo
  # 分组文档
#  docket:
#    user:
#      title: 用户模块
#      base-package:
#    menu:
#      title: 菜单模块
#      base-package:



spring:
  mvc:
    pathmatch:
      matching-strategy: ant_path_matcher



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


    @Autowired
```

```
92      private SwaggerConfigurationProperties swaggerConfigurationProperties;
93
94      private static final Logger log =
LoggerFactory.getLogger(SwaggerAutoConfiguration.class);
95
96      /**
97       * bean工厂
98       */
99      private BeanFactory beanFactory;
100
101     @Override
102     public void setBeanFactory(BeanFactory beanFactory) throws
BeansException
103     {
104         this.beanFactory = beanFactory;
105     }
106
107
108     @Bean
109     @ConditionalOnMissingBean
110     @SuppressWarnings("all")
111     public List<Docket> createRestApi()
112     {
113         ConfigurableBeanFactory configurableBeanFactory =
114                 (ConfigurableBeanFactory) beanFactory;
115         List<Docket> docketList = new LinkedList<>();
116         // 没有分组的情况
117         if (swaggerConfigurationProperties.getDocket().isEmpty())
118         {
119             Docket docket = createDocket(swaggerConfigurationProperties);
120
 configurableBeanFactory.registerSingleton(swaggerConfigurationProperties.g
etTitle(),
121                     docket);
122             docketList.add(docket);
123             return docketList;
124         }
125         // 分组创建
126         for (String groupName :
swaggerConfigurationProperties.getDocket().keySet())
127         {
128             SwaggerConfigurationProperties.DocketInfo docketInfo =
129
 swaggerConfigurationProperties.getDocket().get(groupName);
130             ApiInfo apiInfo = new ApiInfoBuilder()
131                     //页面标题
132                     .title(docketInfo.getTitle())
133                     //创建人
134                     .contact(new Contact
135                             (
136                                     docketInfo.getContact().getName(),
137                                     docketInfo.getContact().getUrl(),
138                                     docketInfo.getContact().getEmail()))
139                     //版本号
140                     .version(docketInfo.getVersion())
141                     //描述
142                     .description(docketInfo.getDescription())
143                     .build();
```

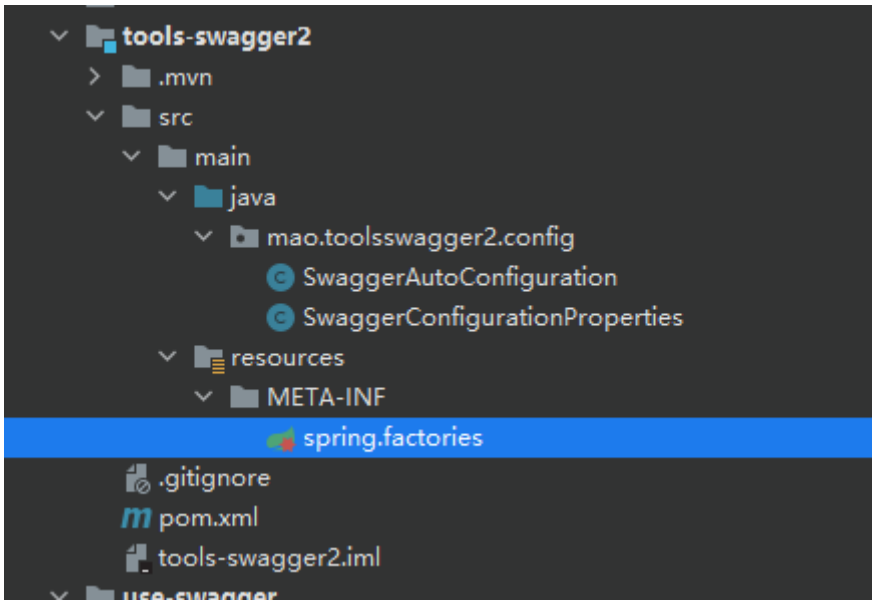
```
144
145             // base-path处理
146             // 当没有配置任何path的时候，解析/**
147             if (docketInfo.getBasePath().isEmpty())
148             {
149                 docketInfo.getBasePath().add("/**");
150             }
151             List<Predicate<String>> basePath = new ArrayList<>();
152             for (String path : docketInfo.getBasePath())
153             {
154                 basePath.add(PathSelectors.ant(path));
155             }
156
157             // exclude-path处理
158             List<Predicate<String>> excludePath = new ArrayList<>();
159             for (String path : docketInfo.getExcludePath())
160             {
161                 excludePath.add(PathSelectors.ant(path));
162             }
163
164             Docket docket = new Docket(DocumentationType.SWAGGER_2)
165                     .apiInfo(apiInfo)
166                     .groupName(docketInfo.getGroup())
167                     .select()
168                     //为当前包路径
169
.apis(RequestHandlerSelectors.basePackage(docketInfo.getBasePackage()))
170
.paths(Predicates.and(Predicates.not(Predicates.or(excludePath)),
Predicates.or(basePath)))
171                     .build();
172             configurableBeanFactory.registerSingleton(groupName, docket);
173             docketList.add(docket);
174         }
175         return docketList;
176     }
177
178     /**
179      * 构建 api文档的详细信息
180      *
181      * @param swaggerConfigurationProperties 配置属性
182      * @return {@link ApiInfo}
183      */
184     private ApiInfo apiInfo(SwaggerConfigurationProperties
swaggerConfigurationProperties)
185     {
186         return new ApiInfoBuilder()
187                 //页面标题
188                 .title(swaggerConfigurationProperties.getTitle())
189                 //创建人
190                 .contact(new Contact
191                         (
192
 swaggerConfigurationProperties.getContact().getName(),
193
 swaggerConfigurationProperties.getContact().getUrl(),
194
 swaggerConfigurationProperties.getContact().getEmail()
```

```
                        )
                )
                //版本号
                .version(swaggerConfigurationProperties.getVersion())
                //描述
.description(swaggerConfigurationProperties.getDescription())
                .build();
    }


    /**
     * 创建接口文档对象
     *
     * @param swaggerConfigurationProperties 配置属性
     * @return {@link Docket}
     */
    @SuppressWarnings("all")
    private Docket createDocket(SwaggerConfigurationProperties
swaggerConfigurationProperties)
    {
        //API 基础信息
        ApiInfo apiInfo = apiInfo(swaggerConfigurationProperties);

        // base-path处理
        // 当没有配置任何path的时候，解析/**
        if (swaggerConfigurationProperties.getBasePath().isEmpty())
        {
            swaggerConfigurationProperties.getBasePath().add("/**");
        }
        List<Predicate<String>> basePath = new ArrayList<>();
        for (String path : swaggerConfigurationProperties.getBasePath())
        {
            basePath.add(PathSelectors.ant(path));
        }

        // exclude-path处理
        List<Predicate<String>> excludePath = new ArrayList<>();
        for (String path : swaggerConfigurationProperties.getExcludePath())
        {
            excludePath.add(PathSelectors.ant(path));
        }

        return new Docket(DocumentationType.SWAGGER_2)
                .apiInfo(apiInfo)
                .groupName(swaggerConfigurationProperties.getGroup())
                .select()
.apis(RequestHandlerSelectors.basePackage(swaggerConfigurationProperties.ge
tBasePackage()))
.paths(Predicates.and(Predicates.not(Predicates.or(excludePath)),
Predicates.or(basePath)))
                .build();
    }


    @PostConstruct
```

```
247     public void init()
248     {
249         log.info("初始化swagger接口文档");
250     }
251
252 }
```

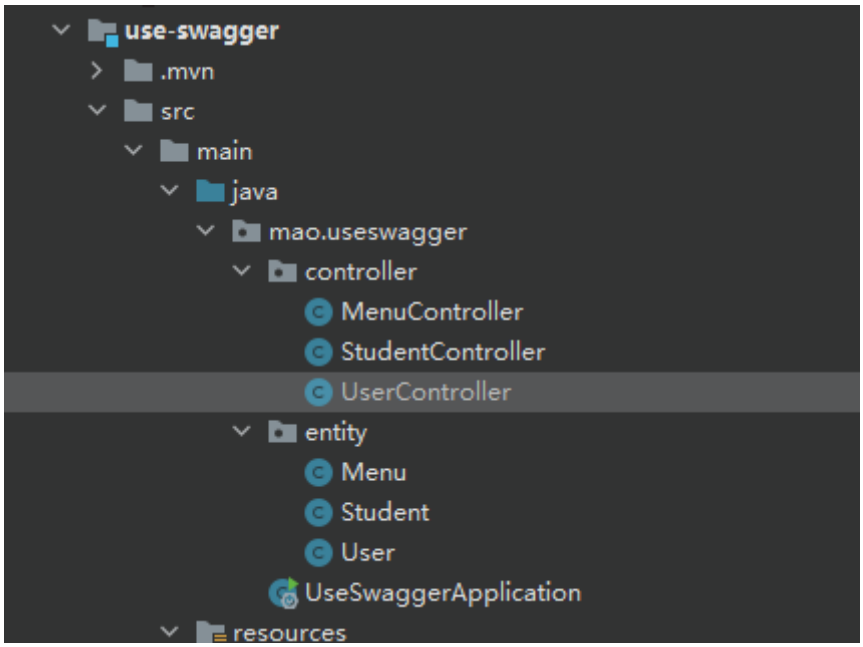
第六步：在resources的META-INF目录下创建spring.factories文件

```
1 org.springframework.boot.autoconfigure.EnableAutoConfiguration=\
2   mao.toolsswagger2.config.SwaggerAutoConfiguration
```



使用starter

第一步：拷贝之前的entity包和controller包到此项目里



第二步：在pom文件中添加tools-swagger2的依赖

```
<?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>
        <artifactId>swagger_starter_demo</artifactId>
        <groupId>mao</groupId>
        <version>0.0.1</version>
    </parent>

    <artifactId>use-swagger</artifactId>
    <version>0.0.1</version>
    <name>use-swagger</name>
    <description>use-swagger</description>

    <properties>

    </properties>

    <dependencies>

        <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>
        </dependency>

        <dependency>
            <groupId>mao</groupId>
            <artifactId>tools-swagger2</artifactId>
            <version>0.0.1</version>
        </dependency>

    </dependencies>

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

</project>
```

第三步：修改配置文件

```
swagger:
  # 是否启用swagger
  enabled: true
  title: 在线文档
  group: 默认组
  version: @version@
  contact:
    name: mao
    url: https://github.com/maomao124/
    email: 1234@qq.com
  base-package: mao.useswagger
  # 分组文档
#  docket:
#    user:
#      title: 用户模块
#      base-package:
#    menu:
#      title: 菜单模块
#      base-package:
```

```yaml
spring:
  mvc:
    pathmatch:
      matching-strategy: ant_path_matcher
```

第四步：启动程序

```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v2.7.1)

2022-10-27 15:35:51.288  INFO 9672 --- [           main] mao.useswagger.UseSwaggerApplication     : Starting UseSwaggerApplication using Java 16.0.2 on mao with PID 9672 (H:\程序\大四上期\swagger_starter_demo\use-swagger\target\classes started by mao in H:\程序\大四上期\swagger_starter_demo)
2022-10-27 15:35:51.291  INFO 9672 --- [           main] mao.useswagger.UseSwaggerApplication     : No active profile set, falling back to 1 default profile: "default"
2022-10-27 15:35:52.240  INFO 9672 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8080 (http)
2022-10-27 15:35:52.249  INFO 9672 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2022-10-27 15:35:52.249  INFO 9672 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Apache Tomcat/9.0.64]
2022-10-27 15:35:52.348  INFO 9672 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2022-10-27 15:35:52.348  INFO 9672 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 1003 ms
2022-10-27 15:35:52.839  INFO 9672 --- [           main] pertySourcedRequestMappingHandlerMapping : Mapped URL path [/v2/api-docs] onto method [springfox.documentation.swagger2.web.Swagger2Controller#getDocumentation(String, HttpServletRequest)]
2022-10-27 15:35:52.848  INFO 9672 --- [           main] m.t.config.SwaggerAutoConfiguration      : 初始化swagger接口文档
2022-10-27 15:35:52.908  INFO 9672 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8080 (http) with context path ''
2022-10-27 15:35:52.909  INFO 9672 --- [           main] d.s.w.p.DocumentationPluginsBootstrapper : Context refreshed
```

```
2022-10-27 15:35:52.919  INFO 9672 --- [           main] d.s.w.p.DocumentationPluginsBootstrapper : Found 1 custom documentation plugin(s)
2022-10-27 15:35:52.943  INFO 9672 --- [           main] s.d.s.w.s.ApiListingReferenceScanner     : Scanning for api listing references
2022-10-27 15:35:52.963  INFO 9672 --- [           main] mao.useswagger.UseSwaggerApplication     : Started UseSwaggerApplication in 1.998 seconds (JVM running for 2.628)
```

第五步：访问

http://localhost:8080/doc.html

Student

名称	类型	说明
age	integer(int32)	学生的年龄
id	integer(int64)	学生的学号，主键
name	string	学生姓名
sex	string	学生性别

User

名称	类型	说明
id	integer(int64)	id，主键
name	string	用户的名字或者昵称
password	string	用户的密码

end

by mao

2022 10 27