**Swagger2-生成RESTful接口**

\* 所有操作都是在tkmapper框架下进行,不了解如何搭建转springboot-fastJson.

# Pom.xml添加依赖

主要swagger用到的两个依赖

|  |
| --- |
| <!-- swagger RESTful API 文档 -->  <dependency>  <groupId>io.springfox</groupId>  <artifactId>springfox-swagger2</artifactId>  <version>2.2.2</version>  </dependency>  <dependency>  <groupId>io.springfox</groupId>  <artifactId>springfox-swagger-ui</artifactId>  <version>2.2.2</version>  </dependency> |

项目架构采用的springboot+tkmapper,具体如何配置项目,参阅Springboot-fastJson文档.

|  |
| --- |
| <?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 http://maven.apache.org/xsd/maven-4.0.0.xsd">  <modelVersion>4.0.0</modelVersion>   <groupId>com.xcy</groupId>  <artifactId>springboot-swagger</artifactId>  <version>0.0.1-SNAPSHOT</version>  <packaging>jar</packaging>   <name>springboot-swagger</name>  <description>Demo project for Spring Boot</description>   <parent>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-parent</artifactId>  <version>2.1.1.RELEASE</version>  <relativePath/> <!-- lookup parent from repository -->  </parent>   <properties>  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>  <java.version>1.8</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>   <!-- swagger restful API 文档 -->  <dependency>  <groupId>io.springfox</groupId>  <artifactId>springfox-swagger2</artifactId>  <version>2.2.2</version>  </dependency>  <dependency>  <groupId>io.springfox</groupId>  <artifactId>springfox-swagger-ui</artifactId>  <version>2.2.2</version>  </dependency>  <!--fastJson-->  <dependency>  <groupId>com.alibaba</groupId>  <artifactId>fastjson</artifactId>  <version>1.2.31</version>  </dependency>  <!--mysql-->  <dependency>  <groupId>mysql</groupId>  <artifactId>mysql-connector-java</artifactId>  <scope>runtime</scope>  </dependency>  <!--mybatis支持-->  <dependency>  <groupId>org.mybatis.spring.boot</groupId>  <artifactId>mybatis-spring-boot-starter</artifactId>  <version>1.3.1</version>  </dependency>  <dependency>  <groupId>tk.mybatis</groupId>  <artifactId>mapper-spring-boot-starter</artifactId>  <version>2.0.1</version>  </dependency>  <!-- 在编译阶段动态生成代码 -->  <dependency>  <groupId>org.projectlombok</groupId>  <artifactId>lombok</artifactId>  <version>1.16.16</version>  </dependency>  </dependencies>   <build>  <plugins>  <!--mybatis自动生成插件-->  <plugin>  <groupId>org.mybatis.generator</groupId>  <artifactId>mybatis-generator-maven-plugin</artifactId>  <version>1.3.6</version>  <configuration>  <configurationFile>src/main/resources/generator/generatorConfig.xml</configurationFile>  <overwrite>true</overwrite>  <verbose>true</verbose>  </configuration>  <dependencies>  <dependency>  <groupId>mysql</groupId>  <artifactId>mysql-connector-java</artifactId>  <version>5.1.46</version>  </dependency>  <dependency>  <groupId>tk.mybatis</groupId>  <artifactId>mapper-generator</artifactId>  <version>1.0.0</version>  </dependency>  </dependencies>  </plugin>  <plugin>  <groupId>org.apache.maven.plugins</groupId>  <artifactId>maven-compiler-plugin</artifactId>  <version>3.1</version>  <configuration>  <source>1.8</source>  <target>1.8</target>  </configuration>  </plugin>  </plugins>  </build>   </project> |

# 编写SwaggerConfig配置类

|  |
| --- |
| package com.xcy.springbootswagger.config;  import org.springframework.context.annotation.Bean; import org.springframework.context.annotation.Configuration; 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;  */\*\*  \** ***@author*** *xcy  \** ***@date*** *2018/12/05 14:44  \** ***@description*** *Swagger2配置类  \** ***@since*** *V1.0.0  \*/* @Configuration @EnableSwagger2 public class Swagger2Config {  @Bean  public Docket createRestApi() {  return new Docket(DocumentationType.*SWAGGER\_2*)  .apiInfo(apiInfo())  .select()  //.apis(RequestHandlerSelectors.withMethodAnnotation(ApiOperation.class)) //这里采用包含注解的方式来确定要显示的接口  .apis(RequestHandlerSelectors.*basePackage*("com.xcy.springbootswagger.controller")) //这里采用包扫描的方式来确定要显示的接口  .paths(PathSelectors.*any*())  .build();  }   private ApiInfo apiInfo() {  return new ApiInfoBuilder()  .title("Swagger helloWorld Doc")  .description("Swagger CRUD 展示文档")  .license("")  .licenseUrl("")  .termsOfServiceUrl("")  .contact(new Contact("晴天", "", "13717736@qq.com"))  .version("1.0")  .build();  }  } |

# 三.编写拦截器,防止swagger-ui静态资源不被拦截

|  |
| --- |
| package com.xcy.springbootswagger.config;  import org.springframework.context.annotation.Configuration; import org.springframework.web.servlet.config.annotation.ResourceHandlerRegistry; import org.springframework.web.servlet.config.annotation.WebMvcConfigurerAdapter;  */\*\*  \** ***@author*** *xcy  \** ***@date*** *2018/12/05 14:50  \** ***@description*** *\** ***@since*** *V1.0.0  \*/* @Configuration public class WebMVCConfig extends WebMvcConfigurerAdapter {  @Override  public void addResourceHandlers(ResourceHandlerRegistry registry) {  registry.addResourceHandler("swagger-ui.html")  .addResourceLocations("classpath:/META-INF/resources/");  registry.addResourceHandler("/webjars/\*\*")  .addResourceLocations("classpath:/META-INF/resources/webjars/");  } } |

# Controller层+Restful风格的接口编写(CRUD)

## 1.Controller编写,使用@ApiOperation注解

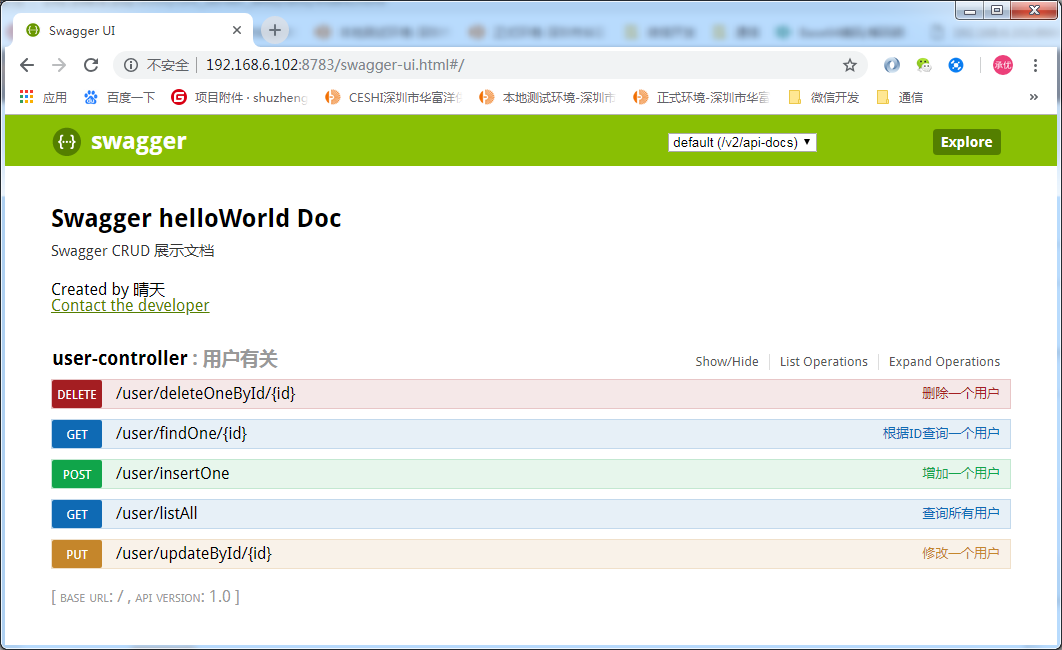
|  |
| --- |
| package com.xcy.springbootswagger.controller;  import com.xcy.springbootswagger.model.User; import com.xcy.springbootswagger.service.UserService; import io.swagger.annotations.Api; import io.swagger.annotations.ApiOperation; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.web.bind.annotation.\*;  import java.util.List;  */\*\*  \** ***@author*** *xcy  \** ***@date*** *2018/12/05 14:22  \** ***@description*** *UserController  \** ***@since*** *V1.0.0  \*/* @Api(description = "用户有关") @RestController @RequestMapping("user") public class UserController {   @Autowired  private UserService userService;   @ApiOperation("查询所有用户")  @GetMapping("listAll")  public List<User> listAll() {  List<User> userList = userService.listAll();  return userList;  }   @ApiOperation("根据ID查询一个用户")  @GetMapping("findOne/{id}")  public User findOneById(@PathVariable Integer id){  User user = userService.findOneById(id);  return user;  }   @ApiOperation("增加一个用户")  @PostMapping("insertOne")  public void insetOne(@ModelAttribute User user){  userService.addOne(user);  }   @ApiOperation("删除一个用户")  @DeleteMapping("deleteOneById/{id}")  public void deleteOneById(@PathVariable Integer id){  userService.deleteOneById(id);  }   @ApiOperation("修改一个用户")  @PutMapping("updateById/{id}")  public void updateById(@PathVariable Integer id){  userService.updateById(id);  }   } |

## 2.实体编写,使用@ApiModelProperty注解

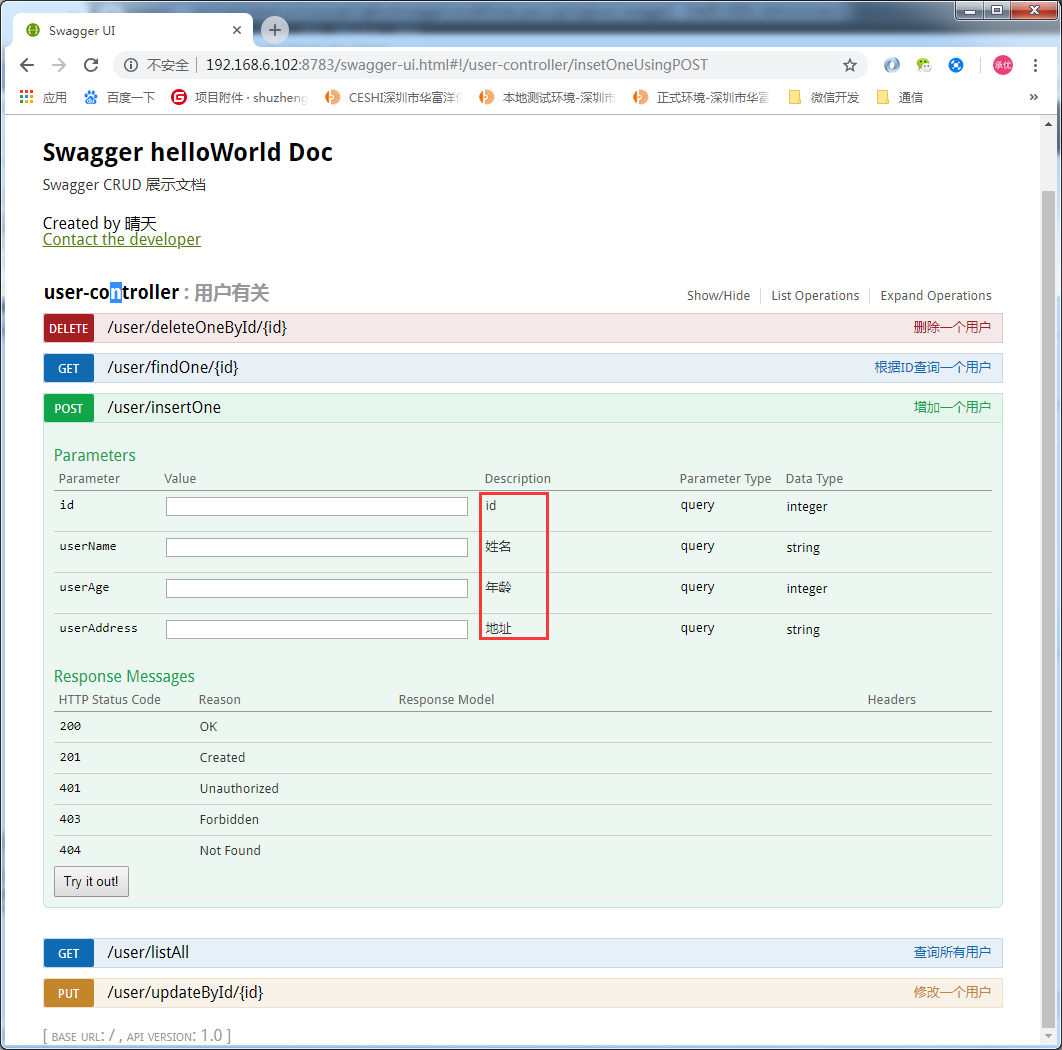
|  |
| --- |
| public class User {  @GeneratedValue(strategy = GenerationType.*IDENTITY*)  @ApiModelProperty(value = "id")  private Integer id;   @Column(name = "user\_name")  @ApiModelProperty(value = "姓名")  private String userName;   @Column(name = "user\_age")  @ApiModelProperty(value = "年龄")  private Integer userAge;   @Column(name = "user\_address")  @ApiModelProperty(value = "地址")  private String userAddress; |

## 3.页面访问swagger

<http://192.168.6.102:8783/swagger-ui.html>



## 4.通过swagger页面调用接口,效果不错.增删改查都实现



@GetMapping("listAll")-->用来查询

@PostMapping("insertOne")-->用来新增

@PutMapping("updateById/{id}")-->用来修改

@DeleteMapping("deleteOneById/{id}")-->用来删除

都是通过使用@RequestMapping(method = RequestMethod.GET)...组合来的