一.需要添加的依赖:(spring-boot:https://start.spring.io/)

1.DecTools(热部署)

2.mysql数据库(选5.1.39)

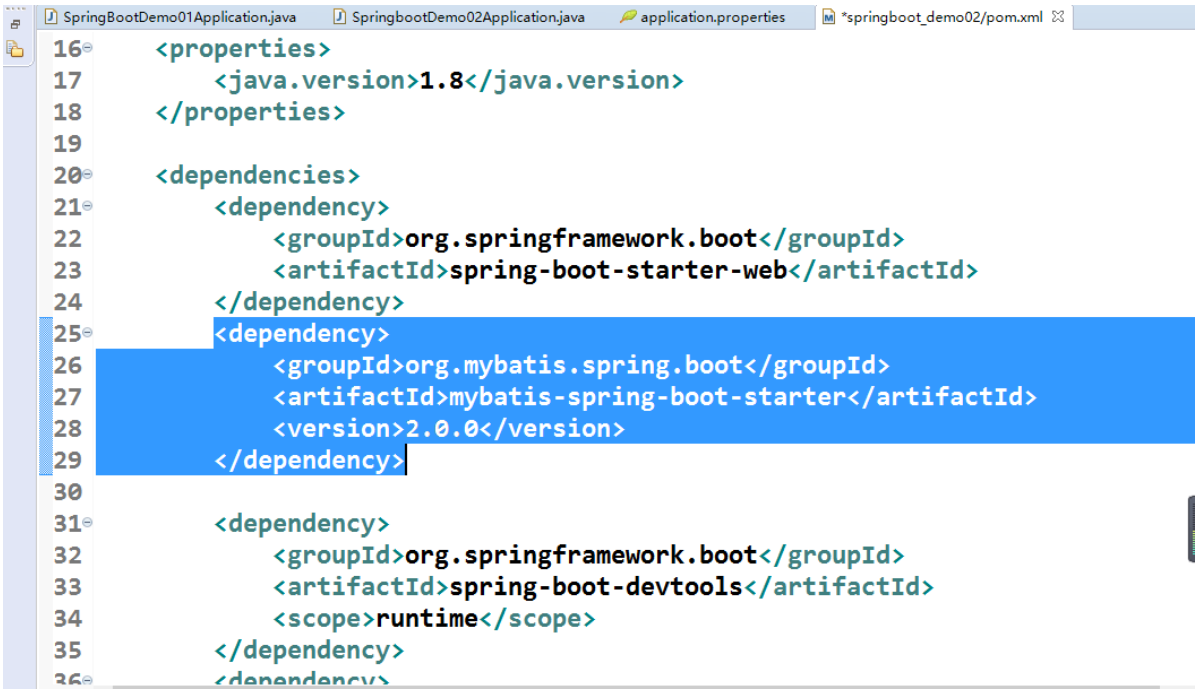
3.myBatis

4.web(view)

5.添加druid(数据源)

6.添加lombok(优雅的编码)

#5.添加pagehelper-spring-boot-starter(分页插件)



二.yml语言

```
1、基本语法
格式要求
```

```
k:(空格)v：表示一对键值对（空格必须有）；
以空格的缩进来控制层级关系；只要是左对齐的一列数据，都是同一个层级的
```

```
server:
 port: 8081
 path: /hello
```

网址:https://blog.csdn.net/qq_30443907/article/details/82706386

```
server:
 port: 9090 #项目的端口号
#数据源的配置
spring:
 datasource:
 type: com.alibaba.druid.pool.DruidDataSource
 username: root
 password: root
 driver-class-name: com.mysql.jdbc.Driver
  url: jdbc:mysql://localhost:3306/shop
  dbcp2:
  min-idle: 5 #进行数据库连接池的配置
  initial-size: 5 #数据库连接池的最小维数
  max-total: 5 #初始化提供的连接数
  max-wait-millis: 200 #等待连接获取的最大超时间

#mybatis的配置
mybatis:
  mapper-locations: classpath:mapping/*.xml #默认去src/main/resources 扫描
  type-aliases-package: com.lanou.springboot_demo02.domain #实体类别名
#显示sql
logging:
 level:
 com.lanou.springboot_demo02.domain: debug
#pagehelper分页插件
pagehelper:
```

```
helperDialect: mysql
reasonable: true
supportMethodsArguments: true
params: count = countSql
returnPageInfo: check
```

三.generatorConfig.xml生成domian和mapping,需要注意的是,resouces是mapping.xml文件,domian是实体类和mapping接口

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE generatorConfiguration PUBLIC "-//mybatis.org//DTD MyBatis Generator Configuration 1.0//EN" "http://mybatis.org/dtd/mybatis-generator-config_1_0.dtd">
<generatorConfiguration>
 <!-- 驱动的绝对位置 -->
 <classPathEntry
 location="E:\Maven\mysql\mysql-connector-java\5.1.39\mysql-connector-java-5.1.39.jar" />
 <context id="context1">
 <!-- 去掉注释 -->
 <commentGenerator>
  <property name="suppressDate" value="true"/>
  <property name="suppressAllComments" value="true" />
  </commentGenerator>
  <jdbcConnection
  connectionURL="jdbc:mysql://localhost:3306/shop"
  driverClass="com.mysql.jdbc.Driver" password="root" userId="root" />

  <javaModelGenerator
  targetPackage="com.lanou.springboot_demo02.domain"
  targetProject="springboot_demo02/src/main/java" />

  <sqlMapGenerator targetPackage="mapping"
  targetProject="springboot_demo02/src/main/resources" />
  <javaClientGenerator
  targetPackage="com.lanou.springboot_demo02.domain"
  targetProject="springboot_demo02/src/main/java"
```

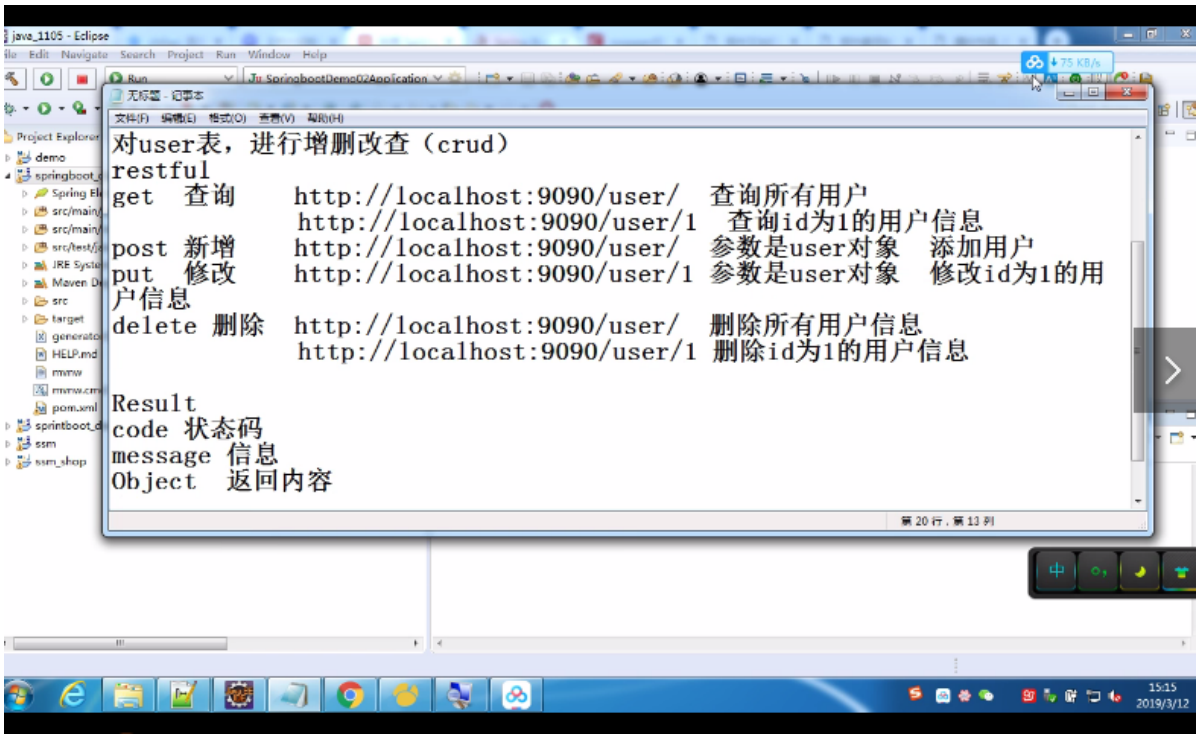
```
  type="XMLMAPPER" />
  <table tableName="smbms_user" domainObjectName="User"
  enableCountByExample="false" enableUpdateByExample="false"
  enableDeleteByExample="false" enableSelectByExample="false"
  selectByExampleQueryId="false">
  <!-- 属性的驼峰的设置 -->
  <property name="useActualColumnNames" value="true" />
  </table>
  </context>
</generatorConfiguration>
```

四.spring-boot　restful

1.需要定义一个Result对象

```
//有优雅的编码:
//1输出:log.Info2@....实体类中的注解

@Getter
@Setter
@AllArgsConstructor
@NoArgsConstructor
public class Result {

  /**
  * 状态码
  */
  private Integer code;

  /**
  * 提示信息
  */
  private String message;

  /**
  * 数据内容
  */
  private Object o;
```

```
24
25  }
```



Swagger(上述只会展示分页的数据(下图),get请求在地址栏可以直接测出来,put,delete,post需要借助swagger)

```
{
    code: 200,
    message: null,
  - o: [
      - {
            id: 1,
            userCode: "admin",
            userName: "张飞",
            userPassword: "123",
            gender: 1,
            birthday: "2019-03-27T16:00:00.000+0000",
            phone: "123454",
            address: "北京",
            userRole: 1,
            createdBy: null,
            creationDate: null,
            modifyBy: null,
            modifyDate: null
        },
      - {
            id: 2,
            userCode: "test",
            userName: "刘能",
            userPassword: "12345",
            gender: 2,
            birthday: "2019-03-19T16:00:00.000+0000",
            phone: "123234",
            address: null,
            userRole: 2,
            createdBy: null,
            creationDate: null,
            modifyBy: null,
            modifyDate: null
        },
      - {
            id: 4,
            userCode: "admin",
            userName: "李白"
```


o[0].createdBy

学习网站:https://blog.csdn.net/sanyaoxu_2/article/details/80555328

运行swagger:http://localhost:9090/swagger-ui.html

模板参数:https://blog.csdn.net/wang78699425/article/details/80572192

查看select代码,log4j:@Slf4j

2.架包

```
<!-- swagger -->
 <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>
 </dependencies>
```

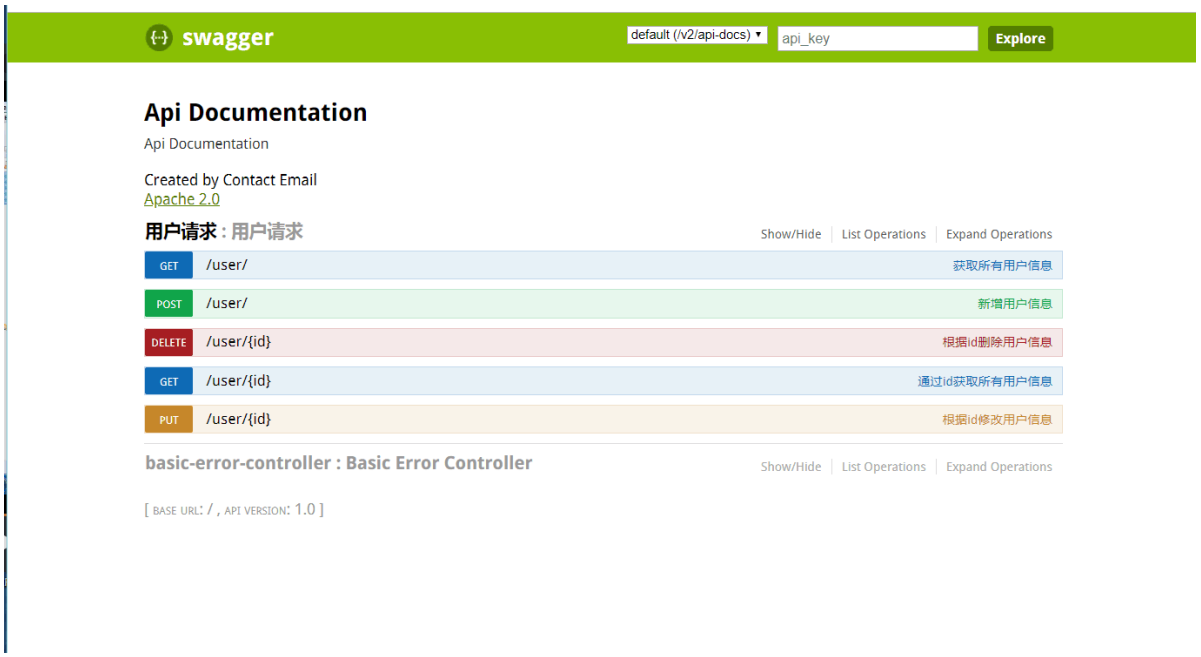
3.在启动类中,添加:(开启swagger)

```
@EnableSwagger2
```

4.controller中,写增删改,注:1UserserviceImpl的实现类中,要使用动态的接口,确保有值传,没值不传.

2.增删改,改需要注意,注意:要传参,id,设置id,修改时id是不能被改的,还需要在@RequestBody中添加@ApiParam自动将Json转为对象

3.@api是写在类中的.@ApiOperation是对每次请求进行说明具体看swagger以上博客.



具体代码:

```
@RestController
@RequestMapping("user")
@Api(value="用户请求")
public class UserController {

```

```
6   @Resource
7   private UserService userService;
8   //说明
9   @ApiOperation(value="获取所有用户信息",notes="获取所有用户信息")
10   @RequestMapping(value="/",method=RequestMethod.GET)
11   public Object getUser(){
12   //当前页内容
13   //get请求在地址栏可以直接测出来,put,delete,post需要借助swagger
14   List<User> users=userService.page(1,10).getList();
15   return new Result(200,null,users);
16
17   }
18   @ApiOperation(value="通过id获取所有用户信息",notes="通过id获取所有用
户信息")
19   @RequestMapping(value="{id}",method=RequestMethod.GET)
20   public Object getUserById(
21   @ApiParam(name="id",value="用户id",required=true)
22   @RequestParam Long id){
23   User u=userService.findUserById(id);
24   return new Result(200,null,u);
25   }
26
27   //传参是json
28   //@RequestBody中添加@ApiParam自动将Json转为对象
29   //简写
30   @ApiOperation("新增用户信息")
31   @PostMapping(value="/")
32   public Object insertUser(
33   @RequestBody @ApiParam(value="用户信息") User u) {
34
35   int row=userService.insert(u);
36   if (row>0) {
37   return new Result(200,null,null);
38   }else {
39   return new Result(500,"添加失败",null);
40
41   }
42
```

```
}
@PutMapping(value="{id}")
@ApiOperation(value="根据id修改用户信息")
public Object update(
@ApiParam(name="id",value="用户Id",required=true)
@RequestParam Long id,
@RequestBody @ApiParam(value="用户信息") User u) {
//获取模板参数
u.setId(id);
int row=userService.updata(u);
if (row>0) {
return new Result(200,null,null);
}else {
return new Result(500,"修改失败",null);
}
}
@DeleteMapping("{id}")
@ApiOperation(value="根据id删除用户信息")
public Object deleteUser(
@ApiParam(name="id",value="用户Id",required=true)
@RequestParam Long id) {
int row=userService.deleteById(id);
if (row>0) {
return new Result(200,null,null);
}else {
return new Result(500,"删除失败",null);
}
}
```