## Post

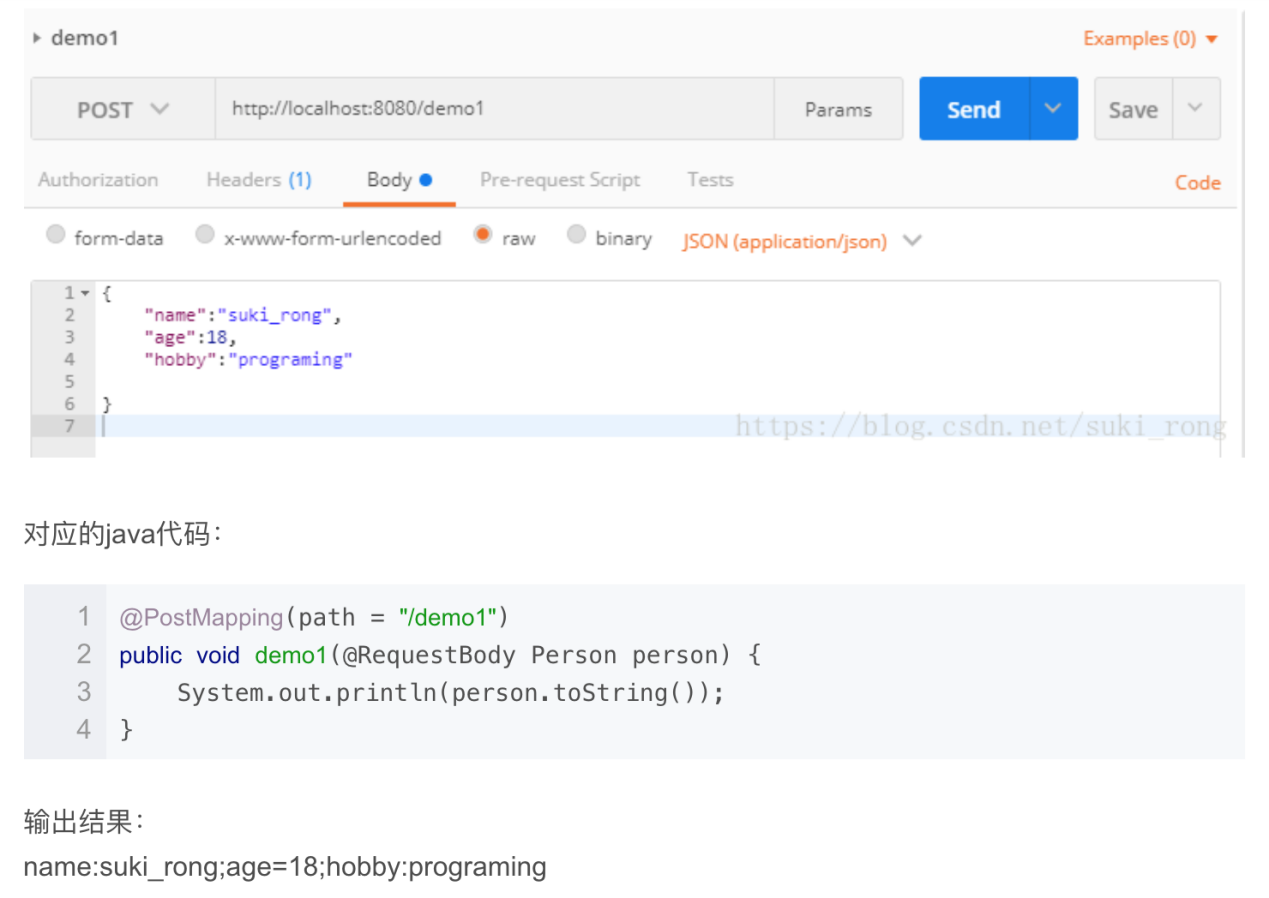
|  |
| --- |
| @Api(value = "PageController", description = "用户登录登出接口")  @Controller  @RequestMapping("/")  public class PageController {  @ApiOperation(value="用户登录", notes="用户登录接口")  @ApiImplicitParams({  @ApiImplicitParam(name = "username", value = "用户名", required = true ,dataType = "string"),  @ApiImplicitParam(name = "passwd", value = "密码", required = true ,dataType = "string")  })  @RequestMapping(value = "/login",method = {RequestMethod.POST,RequestMethod.GET})  @ResponseBody  public ModelMap login(Users data, HttpServletRequest request){  xxx...  }  } |

|  |
| --- |
| @ApiOperation(**"定数包新增"**) @RequestMapping(value = **"v1/add/{materialId}"**, method = {RequestMethod.***POST***, RequestMethod.***GET***}) @ApiImplicitParams({@ApiImplicitParam(name = **"packCoefficient"**, value = **"定数包系数"**, required = **true**, dataType = **"Integer"**) }) **public** RespBean addPacket(  *//实体* @PathVariable(name = **"materialId"**) String materialId,  @RequestBody ImMaterialPackCoefficient imMaterialPackCoefficient ) |
| @ApiOperation(value = "添加景区下的特价门票", notes = "{\"scenicIdArr\" : [53361,53356]}")  @PostMapping(value = "/special\_price/add")  public JSONObject addSpecialPrice(@RequestBody JSONObject scenicIdArr) {    return sprSvc.addSpecialPrice(scenicIdArr);  } |
| @ApiParam(name = **"id"**, value = **"物质ID"**) @PathVariable(name = **"id"**) String id, @ApiParam(name = **"oldPrice"**, value = **"调价后价格"**) @RequestParam String oldPrice, @ApiParam(name = **"reason"**, value = **"调价原因"**) @RequestParam String reason |

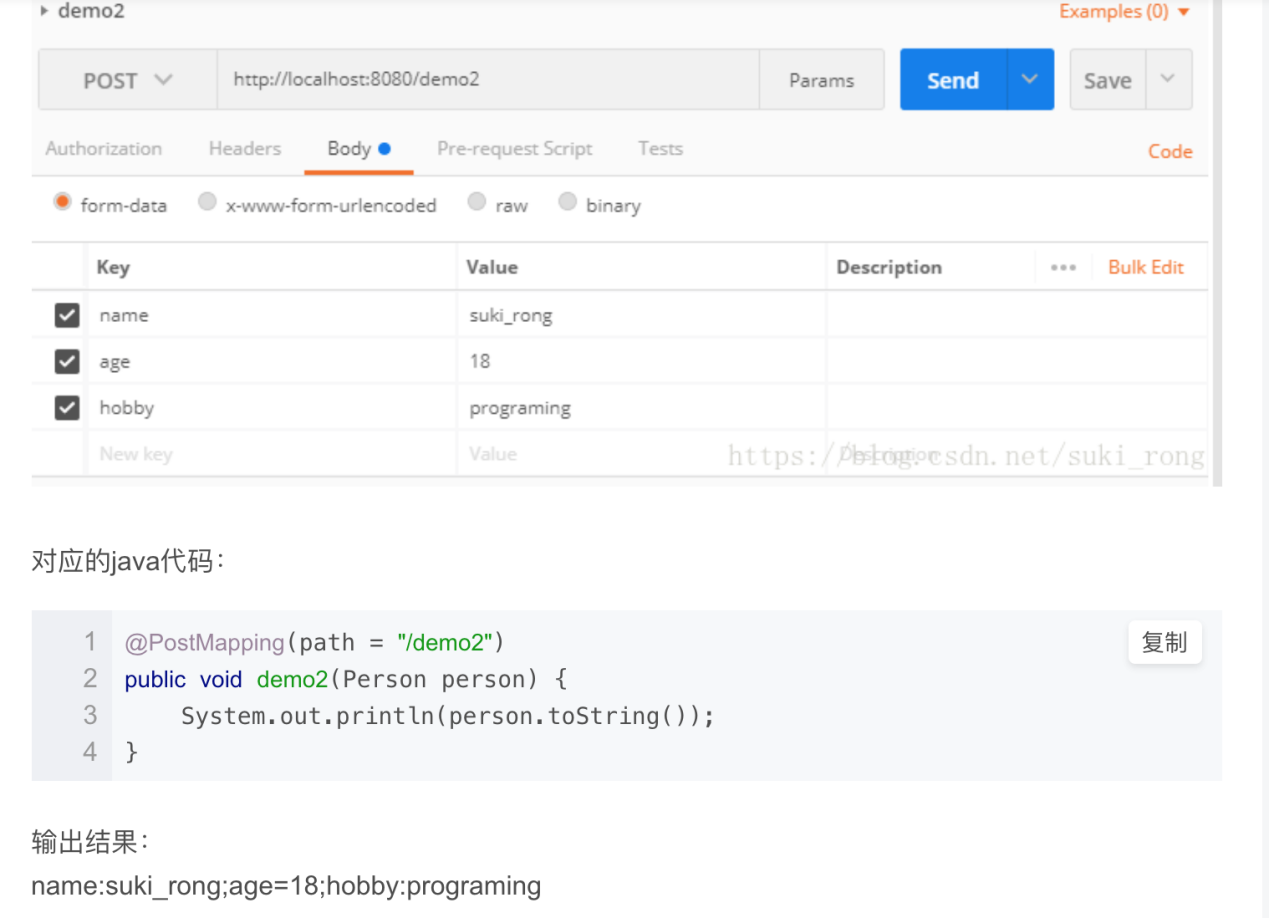
## post传json

|  |
| --- |
| @ApiOperation(**"定数包编辑-保存"**) @RequestMapping(value = **"v1/package/save"**, method = {RequestMethod.***POST***}) @ApiImplicitParams({  @ApiImplicitParam(name = **"id"**, value = **"id"**, required = **true**, dataType = **"Integer"**),  @ApiImplicitParam(name = **"storeCeiling"**, value = **"库存上限"**, required = **true**, dataType = **"Integer"**),  @ApiImplicitParam(name = **"storeFloor"**, value = **"库存下限"**, required = **true**, dataType = **"Integer"**) }) @Transactional **public** RespBean savePackages(  @RequestBody JSONArray jsonArray ) {  ***log***.info(**"接收参数："** + jsonArray);  **if** (jsonArray != **null** && jsonArray.size() > 0) {  List<ImMaterialStorage> list = **new** ArrayList<>(); |















|  |
| --- |
| `end\_time` datetime DEFAULT NULL COMMENT '活动结束时间',  `start\_time` datetime DEFAULT NULL COMMENT '活动开始时间',  DATE\_FORMAT(a.end\_time, '%H:%i:%s' ) **&gt;** DATE\_FORMAT(now(), '%H:%i:%s' )    a.start\_time **&lt;** now()  DATE\_FORMAT(a.start\_time, '%Y-%m-%d %H:%i:%S') start\_time,  IFNULL( a.contract\_type, '' ) AS contract\_type, |
|  |

|  |
| --- |
| mybatis过滤空值  <https://blog.csdn.net/X_Xian_/article/details/77840306> |

## 条件查询

|  |
| --- |
| <select id="select01" resultMap="BasicResultMap">  SELECT  \*  FROM  oa\_employee  <where>  <if test="emp\_name != null and emp\_name != ''">  and emp\_name = #{emp\_name }  </if>  <if test="emp\_sex != null and emp\_sex != ''">  and sex = #{emp\_sex}  </if>  </where>  </select>  <select id="select01" resultMap="BasicResultMap">  SELECT  \*  FROM  oa\_employee  WHERE 1=1  <if test="emp\_name != null and emp\_name != ''">  and emp\_name = #{emp\_name }  </if>  <if test="emp\_sex != null and emp\_sex != ''">  and sex = #{emp\_sex}  </if>  </select> |

|  |
| --- |
| curd分为service和mapper两个地方  带条件查询：  1查询：ImAcceptanceIssues one = issuesService.getOne(new QueryWrapper<ImAcceptanceIssues>().eq("acceptance\_issues\_code","验收问题编码0.8673135946077837"));  2更新： material\_id = iImMaterialStorageService.update(materialStorage, new QueryWrapper<ImMaterialStorage>().eq("material\_id",s)); |

|  |
| --- |
| 分页查询，PageHelper结合mybatis plus使用---还可以过滤删除条件  PageHelper.*startPage*(1, 2); List<ImAcceptanceIssues> list = **issuesService**.list(); PageInfo pageInfo = **new** PageInfo(list); System.***out***.println(pageInfo.toString()); String s = JSONArray.*toJSONString*(list); System.***out***.println(s); |

## 时间比较操作

|  |
| --- |
| newOne.setUpdateTime(LocalDateTime.*now*(ZoneId.*of*(**"UTC+8"**))); newOne.setCreateTime(LocalDateTime.*now*(ZoneId.*of*(**"UTC+8"**)));  **不管时间用什么类型，存正确了就都是一样用的，包括时间比较操作**  @ApiModelProperty(value = **"创建日期"** ) @TableField(value = **"create\_time"**, updateStrategy = ***IGNORED***) @JsonFormat(pattern = **"yyyy-MM-dd HH:mm:ss"**) **private** LocalDateTime **createTime**; |
| `start\_time` datetime DEFAULT NULL COMMENT '起始时间',  `end\_time` datetime DEFAULT NULL COMMENT '结束时间',  AND TIME(a.start\_time) **&lt;**TIME( #{currentTime}) and TIME( #{currentTime}) **&lt;** TIME(a.end\_time);  SimpleDateFormat sdf = **new** SimpleDateFormat(**"HH:mm:ss"**); String current = sdf.format(**new** Date()); Map map = **new** HashMap(); map.put(**"currentTime"**, current); |
| 只比较年月日：  <**if test="endTime!=null and endTime!=''"**>  AND DATE\_FORMAT(a.receipt\_date,'%Y-%m-%d') **&lt;**= DATE\_FORMAT(#{endTime},'%Y-%m-%d') </**if**>  或者是：  .apply(StringUtils.*isNotEmpty*(startTime), **"date\_format(receipt\_date,'%Y-%m-%d') >= {0}"**, startTime.split(**" "**)[0]) .apply(StringUtils.*isNotEmpty*(endTime), **"date\_format(receipt\_date,'%Y-%m-%d') <= {0}"**, endTime.split(**" "**)[0])  年月日时分秒：%Y-%m-%d %H:%i:%s |

## 模糊查询

|  |
| --- |
| <**if test="searchCode !=null and searchCode!=''"**>  and (meas\_unit\_code LIKE '%${searchCode}%' OR meas\_unit\_name LIKE '%${searchCode}%') </**if**> |

## 用list方式传参，使用in查询

|  |
| --- |
| <**select id="getRecordList" resultType="com.bkgtsoft.spd.web.entity.vo.ImMaterialAdjustPriceVO"  parameterType="com.bkgtsoft.spd.web.entity.dto.ImMaterialAdjustPriceRecordDTO"** >   SELECT  b.material\_adjust\_price\_code,  b.apply\_date,  b.apply\_user\_name,  a.material\_name,  a.material\_spce,  a.factory\_name,  a.material\_unit,  a.supply\_name,  a.fee\_code,  a.fee\_flag,  a.medicare\_code,  a.pre\_adjust\_price,  a.adjust\_price  FROM  im\_material\_adjust\_price\_detail a  LEFT JOIN im\_material\_adjust\_price b ON a.main\_id = b.id  <**where**>  <**if test="true"**>  and a.deleted=1  </**if**>  <**if test="materialAdjustPriceCode!=null and materialAdjustPriceCode!=''"**>  and b.material\_adjust\_price\_code like '%${materialAdjustPriceCode}%'  </**if**>  <**if test="startTime!=null and startTime!=''"**>  AND DATE\_FORMAT(b.apply\_date,'%Y-%m-%d') **&gt;**= #{startTime}  </**if**>  <**if test="endTime!=null and endTime!=''"**>  AND DATE\_FORMAT(b.apply\_date,'%Y-%m-%d') **&lt;**= #{endTime}  </**if**>  <**if test="status!=null and status.size()>0"**>  and b.status in  <**foreach collection="status" item="actorId" index="index"  open="(" close=")" separator=","**>  #{actorId}  </**foreach**>  </**if**>  </**where**>  </**select**> |
| @ApiModelProperty(value = **"单据编码"**) **private** String **materialAdjustPriceCode**;  @ApiModelProperty(value = **"申请开始时间，格式2020-09-11 15:25:00"**) **private** String **startTime**;  @ApiModelProperty(value = **"申请结束时间，格式2020-09-11 15:25:00"**) **private** String **endTime**;  @ApiModelProperty(value = **"调价状态：1可调价，2待申报(保存后)，3待审核，4已审核，5审核不通过，6审核通过"**) **private** List<Integer> **status**;  @ApiModelProperty(value = **"申请人"**) **private** String **applyUserName**;  @ApiModelProperty(value = **"申请人ID"**) **private** Integer **applyUserNameId**;  @ApiModelProperty(value = **"物资名称"**) **private** String **materialName**; |

上面用的foreach方式传参， 要注意参数名为【status】

<**if test="status!=null and status.size()>0"**>

## 当前用户

SysUser currentLoginUser = CurrentUserUtil.getCurrentLoginUser();

String userCode = currentLoginUser.getUserCode();

## 查询条件为空值不往后面加

|  |
| --- |
| wrapper.eq(StringUtil.isNotBlank(infoVO.getMobile()), MsPlatformInfo::getPhone, infoVO.getMobile());  当infoVO.getMobile()为空时，StringUtil.isNotBlank(infoVO.getMobile()) = false，这种情况下，此条件在sql执行时不生效  .eq(Integer.*valueOf*(2)!=**null**,**"use\_flag"**, 1)  !StringUtils.*isNullOrEmpty*(**""**),  .eq(!StringUtils.*isNullOrEmpty*(**""**),**"use\_flag"**, 1)  使用不为空判断，不为空就加在后面，为空就不加  .like(StringUtils.*isNotEmpty*(user\_name),**"user\_name"**,user\_name) |

## dto/vo/dao

|  |
| --- |
| ImWarehouseInfoVO imWarehouseInfoVO = **new** ImWarehouseInfoVO(); BeanUtil.*copyProperties*(imWarehouseInfo, imWarehouseInfoVO); **return** imWarehouseInfoVO;  <dependency>  <groupId>cn.hutool</groupId>  <artifactId>hutool-all</artifactId>  <version>5.3.7</version>  </dependency> |

## 获取token

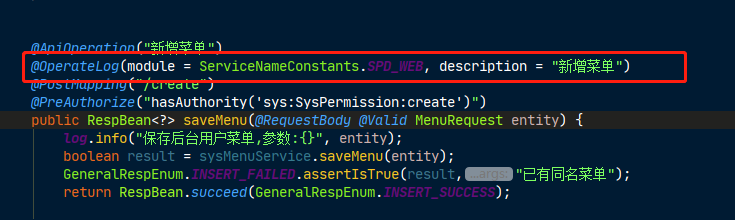
|  |
| --- |
| <http://101.133.217.39:9001/spd-oauth/oauth/token?grant_type=password&username=admin&password=123456&client_secret=secret&client_id=web>  [http://localhost.39:9001/spd-oauth/oauth/token?grant\_type=password&username=admin&password=123456&client\_secret=secret&client\_id=web](http://101.133.217.39:9001/spd-oauth/oauth/token?grant_type=password&username=admin&password=123456&client_secret=secret&client_id=web) |

## LocalDateTime转date

|  |
| --- |
| imMaterialInfo.setUpdateTime(Date.*from*((srmMaterialInfo.getUpdateTime()).atZone(ZoneId.*systemDefault*()).toInstant())); |

## 日志

加上需要写crud日志的@OperateLog



## BigDecimal操作

<https://blog.csdn.net/haibin_hu/article/details/52076830>

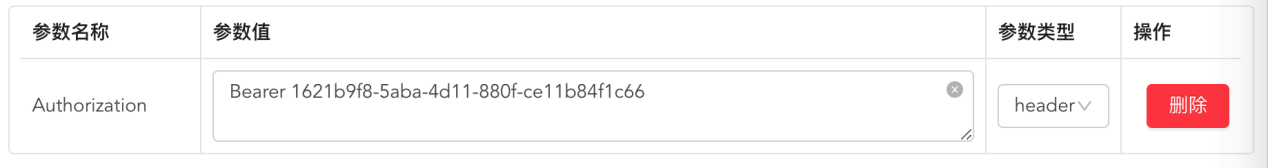
<https://blog.csdn.net/haiyinshushe/article/details/82721234>

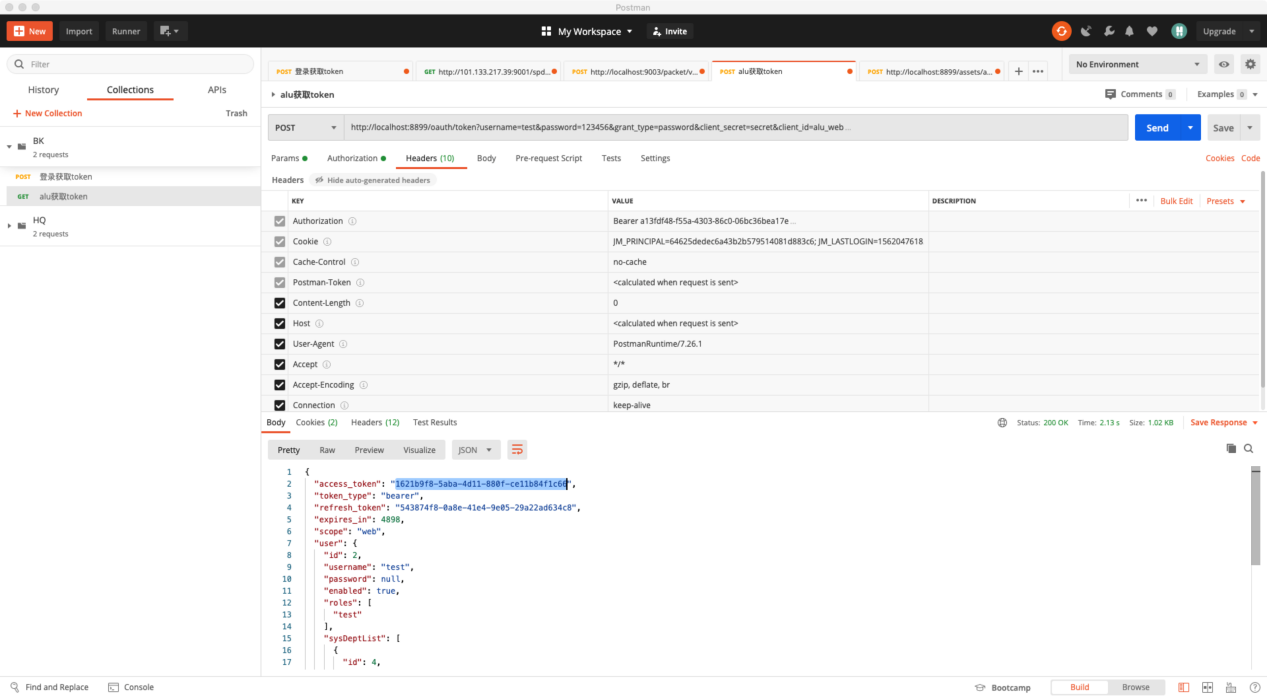
BigDecimal类型除不尽就有异常【比如4/3无限循环就报错】：<https://blog.csdn.net/lopper/article/details/5314686>

BigDecimal divide = payOutlay.divide(giveOutlay, 6, BigDecimal.***ROUND\_DOWN***);  
  
fundOutlay.setRate(divide.setScale(2, BigDecimal.***ROUND\_DOWN***) + **"%"**);

## 权限

Authorization Bearer





## 全局异常

最好用自定义异常，防止用系统的异常，不好进行异常处理。

捕获异常，一定要用@ResponseBody注解

|  |
| --- |
| **package** com.simpleness.config;  **import** org.springframework.web.bind.annotation.ControllerAdvice; **import** org.springframework.web.bind.annotation.ExceptionHandler; **import** org.springframework.web.bind.annotation.ResponseBody;  */\*\*  \** ***@author*** *iszengziqiang@163.com  \** ***@date*** *2020/9/1 18:09  \** ***@Version:*** *1.0  \** ***@desc*** *//****todo*** *\*/* @ControllerAdvice **public class** MyExceptionHandler {   @ResponseBody  @ExceptionHandler(value = Exception.**class**)  **public** String exceptionHandler(Exception e) {  e.printStackTrace();  System.***out***.println(**"未知异常！原因是:"** + e);  **return "全局异常是："** + e.getMessage();  }   @ResponseBody  @ExceptionHandler(value = MyException.**class**)  **public** String myExceptionHandler(MyException e) {  e.printStackTrace();  **return "自定义异常："** + e.getMsg();  }   @ResponseBody  @ExceptionHandler(value = RuntimeException.**class**)  **public** String RuntimeExceptionHandler(RuntimeException e) {  **return "我是运行时异常："** + e.getMessage();  } } |

自定义异常类

|  |
| --- |
| **package** com.simpleness.config;  **import** lombok.Data;  */\*\*  \** ***@author*** *iszengziqiang@163.com  \** ***@date*** *2020/9/11 18:49  \** ***@Version:*** *1.0  \** ***@desc*** *//****todo*** *\*/* @Data **public class** MyException **extends** RuntimeException {   **private** String **msg**;   */\*\*  \* Constructs a new runtime exception with the specified detail message.  \* The cause is not initialized, and may subsequently be initialized by a  \* call to {****@link*** *#initCause}.  \*  \** ***@param message*** *the detail message. The detail message is saved for  \* later retrieval by the {****@link*** *#getMessage()} method.  \*/* **public** MyException(String message, String msg) {  **super**(message);  **this**.**msg** = msg;  }   */\*\*  \* Constructs a new runtime exception with {****@code*** *null} as its  \* detail message. The cause is not initialized, and may subsequently be  \* initialized by a call to {****@link*** *#initCause}.  \*/* **public** MyException(String msg) {  **this**.**msg** = msg;  } } |

抛异常

|  |
| --- |
| @GetMapping(value = **"/error1"**) **public** String error1(**int** a) {  **if** (a == 1) {  **throw new** RuntimeException(**"运行异常啊"**);  }  **return "123"**; } |

## insql和or的使用

|  |
| --- |
| PageHelper.*startPage*(pageNum, pageSize); List<BookClassify> list = list(**new** QueryWrapper<BookClassify>()  .inSql(StringUtils.*isNotEmpty*(classifyStatus), **"classify\_status"**, classifyStatus)  .and(StringUtils.*isNotEmpty*(classifyCode), **new** Consumer<QueryWrapper<BookClassify>>() {  @Override  **public void** accept(QueryWrapper<BookClassify> bookClassifyQueryWrapper) {  bookClassifyQueryWrapper  .like(**"classify\_code"**, classifyCode)  .or()  .like(**"classify\_name"**, classifyCode);  }  })  .orderByDesc(**"update\_time"**) ); PageInfo<BookClassify> pageInfo = **new** PageInfo<>(list); |
| SELECT count(0) FROM book\_classify WHERE (classify\_status IN (1, 2, 4) AND (classify\_code LIKE ? OR classify\_name LIKE ?)) |

## 更新某个字段为null值

|  |
| --- |
| UpdateWrapper<Apply> updateApplyWrapper = **new** UpdateWrapper(); updateApplyWrapper.eq(**"id"**, apply.getId()); **if** (apply.getTemplateId() == **null** || apply.getTemplateId() == 0) {  updateApplyWrapper.set(**"template\_id"**, **null**); } **boolean** update = update(apply, updateApplyWrapper);  **if** (!update) {  **throw new** HandException(-1, **"申请单更新失败"**); } |

自己构建一个更新条件，查看某个字段是否需要更新，apply的值正常更新，**template\_id**会更新为null。

第二种方式：这样也能生效

|  |
| --- |
| */\*\*  \* 字段验证策略之 update: 当更新操作时，该字段拼接set语句时的策略  \* <p>  \* IGNORED: 直接拼接 update table\_a set column=#{columnProperty}, 属性为null/空string都会被set进去  \* NOT\_NULL: update table\_a set <if test="columnProperty != null">column=#{columnProperty}</if>  \* NOT\_EMPTY: update table\_a set <if test="columnProperty != null and columnProperty!=''">column=#{columnProperty}</if>  \* NOT\_EMPTY 如果针对的是非 CharSequence 类型的字段则效果等于 NOT\_NULL  \*  \** ***@since*** *3.1.2  \*/* FieldStrategy updateStrategy() **default** FieldStrategy.***DEFAULT***;  */\*\*  \* 开户到期日期  \*/* @TableField(updateStrategy = FieldStrategy.***IGNORED***) **private** LocalDate **openLicenseExpireDate**; |

## 序列化过滤掉null值

|  |
| --- |
| @Data @TableName(**"acceptance"**) @JsonInclude(JsonInclude.Include.***NON\_NULL***) @ApiModel(value = **"Acceptance对象"**, description = **"验收表"**) **public class** Acceptance **implements** Serializable { |

## 捕获索引重复异常

|  |
| --- |
| @ExceptionHandler(value = DataAccessException.class)  @ResponseBody  public RespJson<Void> repeatException(SQLIntegrityConstraintViolationException e) {  return RespJson.fail(CommonErrorCode.DATA\_ALREADY\_EXISTS);  } |

## 统一返回类型

|  |
| --- |
| **package** com.zq.result;  **import** lombok.Data;  @Data @JsonInclude(JsonInclude.Include.NON\_NULL)*//过来null值* **public class** ResponseResult<T> {   **private** Integer **status**;*//状态值* **private boolean success**;  **private** String **msg**;*//提示* **private** T **data**;*//数据* **public** ResponseResult(**boolean** success, String msg, T data) {  **this**.**success** = success;  **this**.**msg** = msg;  **this**.**data** = data;  }   **public** ResponseResult(Integer status, **boolean** success, String msg, T data) {  **this**.**status** = status;  **this**.**success** = success;  **this**.**msg** = msg;  **this**.**data** = data;  }   **public** ResponseResult(**boolean** success, String msg) {  **this**.**success** = success;  **this**.**msg** = msg;  }   **public** ResponseResult(**boolean** success, T data) {  **this**.**success** = success;  **this**.**data** = data;  }   **public static** <T> ResponseResult<T> success(String msg, T data) {  **return new** ResponseResult(**true**, msg, data);  }   **public static** <T> ResponseResult<T> success(T data) {  **return new** ResponseResult(**true**, data);  }   **public static** <T> ResponseResult<T> success(ResultStatusEnum resultStatusEnum, T data) {  **return new** ResponseResult(resultStatusEnum.getStatus(), **true**, resultStatusEnum.getMsg(), data);  }   **public static** <T> ResponseResult<T> error(ResultStatusEnum resultStatusEnum) {  **return new** ResponseResult(resultStatusEnum.getStatus(), **false**, resultStatusEnum.getMsg(), **null**);  }   **public static** <T> ResponseResult<T> error(String msg) {  **return new** ResponseResult(**false**, msg);  }   } |
| **package** com.zq.result;  **import** lombok.Getter;  @Getter **public enum** ResultStatusEnum {  */\*\*  \* 广义通用  \*/* ***INSERT\_SUCCESS***(2002, **"新增成功"**),  ***INSERT\_FAILED***(3002, **"新增失败"**),   ***DELETE\_SUCCESS***(2004, **"删除成功"**),  ***DELETE\_FAILED***(3004, **"删除失败"**),   ***UPDATE\_SUCCESS***(2003, **"更新成功"**),  ***UPDATE\_FAILED***(3003, **"更新失败"**),   ***QUERY\_SUCCESS***(2001, **"查询成功"**),  ***QUERY\_FAILED***(3001, **"查询失败"**),   ***EXECUTE\_SUCCESS***(205, **"操作成功"**),  ***EXECUTE\_FAILED***(305, **"操作失败"**),  ***BATCH\_UPDATE\_FAILED***(304, **"批量更新失败"**),  ***PARAMETER\_IS\_NULL***(305, **"参数为空"**),  ***PARAMETER\_FAILED***(306, **"无效参数"**),   */\*\*  \* 规约：时间异常类 value定义区间为 -100 ~ -199  \*/* ***TIME\_EMPTY***(-100, **"时间不能为空"**),  ***TIME\_MORE\_THAN\_14\_DAYS***(-101, **"时间跨度不能超过两个星期"**),   */\*\*  \* 规约：字段为空 value区间 -200 ~ -299  \*/* ***OPENID\_EMPTY***(-201, **"openid不能为空"**);   **private int status**;  **private** String **msg**;   ResultStatusEnum(**int** status, String msg) {  **this**.**status** = status;  **this**.**msg** = msg;  } } |

## 使用mybatis-plus乐观锁

**总结：只有在实体类上设值才有效果**

**version如果不对，不会报错，但更新条数为0**

|  |
| --- |
| 引入配置：  @Bean **public** OptimisticLockerInterceptor optimisticLoker() {  **return new** OptimisticLockerInterceptor(); }  字段加注解：@Version  **import** com.baomidou.mybatisplus.annotation.Version; |

### 使用1：OK

查询多少就传入多少：

|  |
| --- |
| Integer id = 7; User user = **userMapper**.selectById(id); System.***out***.println(user.toString()); user.setAddress(**"更新后："** + id); **int** i = **userMapper**.updateById(user); System.***out***.println(**"更新后结果："** + i);  2020-12-18 13:45:26.477 INFO 13173 --- [ main] com.alibaba.druid.pool.DruidDataSource : {dataSource-1} inited  2020-12-18 13:45:26.490 DEBUG 13173 --- [ main] com.xxxxdb.mapper.UserMapper.selectById : ==> Preparing: SELECT id,name,address,create\_time,update\_time,version FROM user WHERE id=?  2020-12-18 13:45:26.915 DEBUG 13173 --- [ main] com.xxxxdb.mapper.UserMapper.selectById : ==> Parameters: 7(Integer)  2020-12-18 13:45:26.956 DEBUG 13173 --- [ main] com.xxxxdb.mapper.UserMapper.selectById : <== Total: 1  User(id=7, name=姓名：10, address=地址：10, createTime=null, updateTime=2020-12-15T10:40:35, version=1)  2020-12-18 13:45:26.996 DEBUG 13173 --- [ main] com.xxxxdb.mapper.UserMapper.updateById : ==> Preparing: UPDATE user SET name=?, address=?, update\_time=?, version=? WHERE id=? AND version=?  2020-12-18 13:45:27.015 DEBUG 13173 --- [ main] com.xxxxdb.mapper.UserMapper.updateById : ==> Parameters: 姓名：10(String), 更新后：7(String), 2020-12-15T10:40:35(LocalDateTime), 2(Integer), 7(Integer), 1(Integer)  2020-12-18 13:45:27.021 DEBUG 13173 --- [ main] com.xxxxdb.mapper.UserMapper.updateById : <== Updates: 1  更新后结果：1 |

### 使用2：x【不加入version的情况】

|  |
| --- |
| User u = **new** User(); u.setId(7); u.setName(**"我更新了名称"**); **int** i = **userMapper**.updateById(u); System.***out***.println(**"更新后结果："** + i);  **由此可见：不加入version不影响操作，只是没有加入version语句，数据库的version也不会变**  2020-12-18 13:52:08.759 INFO 13464 --- [ main] com.alibaba.druid.pool.DruidDataSource : {dataSource-1} inited  2020-12-18 13:52:08.771 DEBUG 13464 --- [ main] com.xxxxdb.mapper.UserMapper.updateById : ==> Preparing: UPDATE user SET name=? WHERE id=?  2020-12-18 13:52:09.205 DEBUG 13464 --- [ main] com.xxxxdb.mapper.UserMapper.updateById : ==> Parameters: 我更新了名称(String), 7(Integer)  2020-12-18 13:52:09.212 DEBUG 13464 --- [ main] com.xxxxdb.mapper.UserMapper.updateById : <== Updates: 1  更新后结果：1 |

### 使用3：x

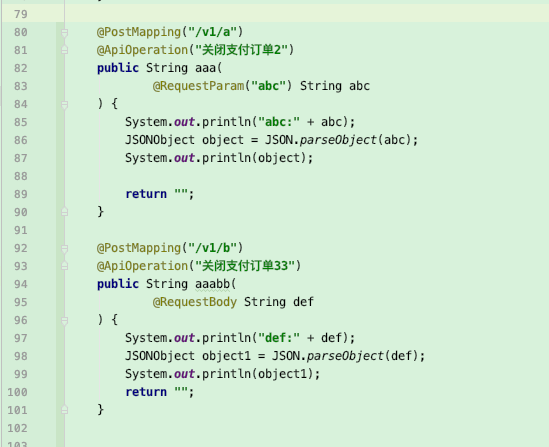
|  |
| --- |
| User u = **new** User(); u.setName(**"我更新了名称8"**); **int** i = **userMapper**.update(u, **new** UpdateWrapper<User>().eq(**"id"**, 8).eq(**"version"**, 1)); System.***out***.println(**"更新后结果："** + i);  **这种情况的version也没有起作用，相对与一个更新条件，自身并没有改变值，没有update**  2020-12-18 13:58:23.225 INFO 13711 --- [ main] com.alibaba.druid.pool.DruidDataSource : {dataSource-1} inited  2020-12-18 13:58:23.256 DEBUG 13711 --- [ main] com.xxxxdb.mapper.UserMapper.update : ==> Preparing: UPDATE user SET name=? WHERE (id = ? AND version = ?)  2020-12-18 13:58:23.976 DEBUG 13711 --- [ main] com.xxxxdb.mapper.UserMapper.update : ==> Parameters: 我更新了名称8(String), 8(Integer), 1(Integer)  2020-12-18 13:58:23.983 DEBUG 13711 --- [ main] com.xxxxdb.mapper.UserMapper.update : <== Updates: 1  更新后结果：1 |

### 使用4：OK

|  |
| --- |
| User u = **new** User(); u.setName(**"我更新了名称9"**); u.setVersion(1); **int** i = **userMapper**.update(u, **new** UpdateWrapper<User>().eq(**"id"**, 9)); System.***out***.println(**"更新后结果："** + i);  这种情况version生效  2020-12-18 14:03:01.192 INFO 13877 --- [ main] com.alibaba.druid.pool.DruidDataSource : {dataSource-1} inited  2020-12-18 14:03:01.204 DEBUG 13877 --- [ main] com.xxxxdb.mapper.UserMapper.update : ==> Preparing: UPDATE user SET name=?, version=? WHERE (id = ? AND version = ?)  2020-12-18 14:03:01.577 DEBUG 13877 --- [ main] com.xxxxdb.mapper.UserMapper.update : ==> Parameters: 我更新了名称9(String), 2(Integer), 9(Integer), 1(Integer)  2020-12-18 14:03:01.583 DEBUG 13877 --- [ main] com.xxxxdb.mapper.UserMapper.update : <== Updates: 1  更新后结果：1 |

# @RequestParam("abc")与@RequestBody区别

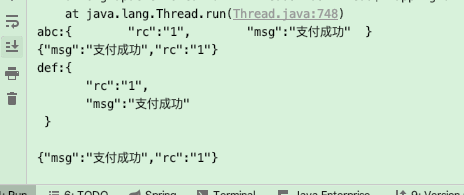
两个接口



requestParam方式：



后台显示



requestBody方式



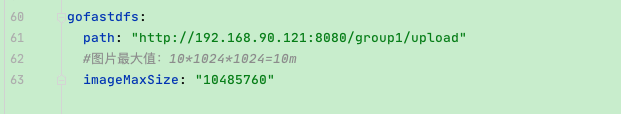
都可以用于String的传递，如果是对象就需要用requestBody方式

# page对象互转

|  |
| --- |
| @Override **public** Page<SupplyInfoVO> customList(String supplierName, Boolean enabled, Integer pageNum, Integer pageSize) {   LambdaQueryWrapper<ImSupplier> queryWrapper = **new** LambdaQueryWrapper<>();  queryWrapper.select(ImSupplier::getId, ImSupplier::getSupplierName, ImSupplier::getContactPerson,  ImSupplier::getContactPhone, ImSupplier::getSupplierCode  , ImSupplier::getEnabled, ImSupplier::getAddress)  .eq(ObjectUtil.*isNotEmpty*(enabled), ImSupplier::getEnabled, enabled)  .like(!StringUtils.*isEmpty*(supplierName), ImSupplier::getSupplierName, supplierName).or()  .like(!StringUtils.*isEmpty*(supplierName), ImSupplier::getSupplierCode, supplierName).or()  .like(!StringUtils.*isEmpty*(supplierName), ImSupplier::getFiveStrokeCode, supplierName).or()  .like(!StringUtils.*isEmpty*(supplierName), ImSupplier::getPinyinCode, supplierName)  .orderByDesc(ImSupplier::getUpdateTime);  Page<ImSupplier> page = **this**.page(**new** Page<>(pageNum, pageSize), queryWrapper);  Page<SupplyInfoVO> result = **new** Page<>();  BeanUtil.*copyProperties*(page, result, **"records"**);  List<ImSupplier> records = page.getRecords();  List<SupplyInfoVO> list = records.parallelStream().map(**this**::toVO).collect(Collectors.*toList*());  result.setRecords(list);  **return** result; }  //方法  **private** SupplyInfoVO toVO(ImSupplier po) {  SupplyInfoVO vo = **new** SupplyInfoVO();  BeanUtil.*copyProperties*(po, vo);  **if** (po.getEnabled()) {  vo.setEnabledStr(**"启用"**);  } **else** {  vo.setEnabledStr(**"停用"**);  }  **return** vo; } |

# 引入自定义yml中的值

yml:



定义一个类：



引用：



使用：

