package com.geekplus.athena.common.entity.api.msg;  
  
import java.io.Serializable;  
  
import lombok.AllArgsConstructor;  
import lombok.Builder;  
import lombok.Getter;  
import lombok.NoArgsConstructor;  
import lombok.Setter;  
  
import com.fasterxml.jackson.annotation.JsonIgnore;  
import com.fasterxml.jackson.annotation.JsonInclude;  
import com.fasterxml.jackson.annotation.JsonInclude.Include;  
  
import com.geekplus.athena.api.APIProtocolType;  
  
*/\*\*  
 \* 回调请求头  
 \*  
 \** ***@author*** *liujun 2018年11月20日  
 \** ***@since*** *3.1.0  
 \*/*@Getter  
@Setter  
@Builder  
@NoArgsConstructor  
@AllArgsConstructor

//javaobject与代表json互转时，为null得不参与序列化

JSONObject msg=(JSONObject) JSONObject.*toJSON*(robotTaskRequestMsg);

及当实体属性为null时，不转化为json字段  
@JsonInclude(value = Include.*NON\_NULL*)

//序列化务必实现Serializable接口  
public class CallbackHeader implements Serializable {  
 private static final long *serialVersionUID* = 1L;  
 */\*\*  
 \* 客户端标识。RMS根据clientCode去读取用户的相关配置（如语言、版本、认证信息等）  
 \* （RMS内部字段，仅用于获取链路通道）  
 \*  
 \** ***@since*** *3.1.0  
 \*\*/* private String clientCode;  
 */\*\*  
 \* 此回调所对应的请求消息，对应{****@link*** *RequestHeader}中的#requestId字段  
 \* （RMS内部字段，用于跟踪消息处理的生命周期）  
 \*  
 \** ***@since*** *3.1.0  
 \*\*/*

*//@JsonIgnore在JSON.toJson (stuedent.class)序列化时，不转化这个属性，比如这个代表转化后的json没有cycleId这个字段*

@JsonIgnore  
 private String cycleId;  
 */\*\*  
 \* 链路通道的标识，用于标识消息的链路来源。  
 \* 一个ClientCode下可以有多条链路（多socket/http/rpc客户端同时连接RMS），但每个通道的channelId必须唯一。  
 \* <p>  
 \* （RMS内部字段，仅用于获取链路通道）  
 \*  
 \** ***@since*** *3.1.0  
 \*\*/* @JsonIgnore  
 private String channelId;  
  
 */\*\*  
 \* 消息来源  
 \* （RMS内部字段，仅用于选择回调实现）  
 \*  
 \** ***@since*** *3.1.0  
 \*\*/* @JsonIgnore  
 private APIProtocolType sourceType;  
  
  
 */\*\*  
 \* 仓库唯一标识  
 \*  
 \** ***@since*** *3.1.0  
 \*\*/* private String warehouseCode;  
  
 */\*\*  
 \* <p>一次请求的唯一标识，有两个作用：</p>  
 \* <p>1. 在异步通信协议中，必须有一个唯一的请求id才能将请求、响应和回调消息对接起来。</p>  
 \* <p>2. 外部系统根据该id判断是否出现重复提交</p>  
 \*  
 \** ***@since*** *3.1.0  
 \*\*/* private String requestId;  
  
 */\*\*  
 \* API版本  
 \*  
 \** ***@since*** *3.1.0  
 \*\*/* private String version;  
  
 */\*\*  
 \* 透传字段（以原样返回用户传输的数据）  
 \*  
 \** ***@since*** *3.1.0  
 \*/* private String ext;  
  
 */\*\*  
 \* 记录来源消息类型，用于处理长短消息（RMS内部字段）  
 \*  
 \** ***@since*** *3.2.0  
 \*/* @JsonIgnore  
 private String requestMsgType;  
  
}

反序列化

public class MapCell {  
// Map<String,Object> mapCelllist;  
  
 Map<String,String> index;  
 Map<String,String> location;  
 String cellType;  
 String cellStatus;  
 Object direction;  
 Object dirUnload;  
 Object allocatedRobotId;  
 Object occupyRobotId;  
 Object occupiedShelfCode;  
 boolean a;  
 int b;  
  
  
 public static void main(String[] args) {  
 MapCell mapCell=new MapCell();  
 MapCell json2mapCell=new MapCell();  
 JSONObject jsonObject1 = new JSONObject();  
 jsonObject1.put("cellType","shelf\_cell");

1jsonobjet转 实体  
 json2mapCell= jsonObject1.toJavaObject(MapCell.class);

反序列化时实体属性只有赋值的有值，其他为null或0或fasle

2 jsonstring转换为实体，包括jsonobject实体

JSON.*parseObject*(jsonObject1.toJSONString(),JSONObject.class);

3,实体转json:注意类型为javaobjet不用带.Class

JSONObject jsonObject = (JSONObject)JSONObject.*toJSON*(mapCell);