1. 解析response，必须填写roomName、roomIndex、totalRate、roomRateId、policies、meal、

2、小孩转成人问题相关。

<1>比如我们前端搜索的条件是1大人1小孩，crawler端可能会转化成两个大人去搜索房源，这时候可能拿到的报价就是两个大人的报价。所以下单的时候要把小孩转化为成人，然后再去填充这个报价。

<2>也有其他情况，比如2个大人要住triple，crawler用的是3个大人去搜索，并且下单的时候gds要求triple必须传3个人的名字，这时候缺了一个名字，那就copy一份大人的名字过去。

<3>还有其他情况，1大人2小孩要住double，crawler转化成2大人一小孩去搜索，我们下单的时候也要转化成2大人一小孩去下单，这时候有两个小孩，我们选择把年龄大的那个小孩转化为成人。

总的来说，就是下单需求的人数不够，先用小孩中年龄大的补充，如果所有小孩都用完了，还没有达到下单需求的人数，则复制成人的名字来进行填充。

3、房型合并的情况的处理。

默认情况下是在rewriteRoom中根据roomType进行房型合并，roomIndex以逗号分隔 1,2,3

如果supplier接受房型合并请求，则不需要拆分处理。

如果supplier不接受房型合并请求，则需要拆分处理。

有两种方式处理，一种是在合并前，覆盖roomCombineKey来实现。

crawler.supplier.base.HotelAvailabilityMultiRequestRewriter

roomCombineKey();

使用方式见Tavisca

@Override

protected String roomCombineKey(Room room, int index) {

return room.getRoomIndex();

}

另一种是在request中调用

crawler.supplier.base.HotelAvailabilityRequestRewriter 中提供的separateRooms 进行房型拆分。

使用方式见miki7

List<Room> roomsOrigin = avRequest.getRooms();

List<Room> roomList = HotelAvailabilityRequestRewriter.separateRooms(roomsOrigin);

4、

对Rewrite功能的梳理：

1. 父类crawler.services.HotelAvailabilityRequestRewriter
   1. 单酒店rewrite

根据hotelmap将request中的countryid、cityid、hotelid更换为countryCode、cityCode、hotelCode。

* 1. 城市rewrite

根据citymap将request中的countryid、cityid更换为countryCode、cityCode。

* 1. 提供基础rewriteRooms

将rooms进行clone，返回List<Rooms>

* 1. 提供combineRooms，将房型进行组合

入：List<Rooms>

例：Rooms

List<Room> Room(2,1) Room(3)

Rooms

List<Room> Room(2) Room(1,1)

Rooms

List<Room> Room(3)

中间状态：List<Rooms>

例：Rooms

List<Room> Room(2,1) Room(2) Room(3)

Rooms

List<Room> Room(2,1) Room(1,1) Room(3)

Rooms

List<Room> Room(3) Room(2) Room(3)

Rooms

List<Room> Room(3) Room(1,1) Room(3)

输出：

如果没有选择roomType

返回值为中间状态值

如果选择了roomType

Key:3doubledoubleTriple Room(2,1) Room(2) Room(3)

Key:3doubledoubleTriple Room(2,1) Room(1,1) Room(3)

Key:3TripledoubleTriple Room(3) Room(2) Room(3)

Key:3TripledoubleTriple Room(3) Room(1,1) Room(3)

结果：

Rooms Room(2,1) Room(2) Room(3)

--indexes = 1,2

Rooms Room(3) Room(2) Room(3)

--index= 1,2

解释：客户想要三间房，第一间要住2A1C，可以rewrite为一间2A1C和一间3A. 那么就会增加一个房型匹配。一般的，方法使用前应该会调用一个房型rewrite的方法。比如travco覆盖的rewriteSimilarRoom方法。

* 1. 提供mergeRoom0方法：

根据adults与childs是否相等进行合并，合并处理roomIndex与ages。

roomIndex:1,2,3

ages:4,5,6

* 1. isRequestRoomNumValid

判断是否超过了设置的最大房间数。

注意：必须设置supplier.rewrite.maxRoomNum来实现控制，否则无效。

1. 一级子类crawler.supplier.base.HotelAvailabilityRequestRewriter
   1. 覆盖rewriteRooms

根据roomKey的不同，实现了对room中rooms、roomIndex的合并，合并方式依赖于room对象内部的赋值方式。

* 1. separateRooms新增方法

roomIndex:"1,2" 拆成2间房

何时将rooms合并的？何时需要拆分？

* 1. roomKey新增方法

roomKey = **"Room(roomType="** + roomType + **", adults="** + room.getAdults() + **", children="** + room.getChildren() + **", ages="** + ages + **")"**;

3、二级子类

HotelAvailabilityMultiRequestRewriter

覆盖rewriteRooms

--toGdsGuest

--getPaxType

--validateRoom

--rewriteSimilarRoom

--combineRooms

--simplifyRoomList(与父类的mergeRoom稍有不同)

--roomCombineKey

功能解析：

rewriteRooms：

对room进行房型组合(一般的需要先将room进行rewrite SimilarRoom，可以追踪rewriteSimilarRoom查看相关实现)，并将相同房间根据roomCombineKey进行归并。

toGdsGuest：没实现功能，用来被覆盖。

设计用来实现对成人与儿童的转换。可根据不同gds要求将儿童转换为成人。

validateRoom：没实现功能，用来被覆盖

设计用来实现对单room的验证过滤。

rewriteSimilarRoom：没实现功能，用来被覆盖。

根据不同的gds要求，控制一个room可以rewrit为几个room。simplifyRoomList合并room，规则依赖roomCombineKey实现

rooms(roomCount)求和

合并room的children(儿童数量)取最大值。

合并room的Index以逗号分隔。

roomCombineKey

默认以roomType进行房型标识。可用来覆盖

validateRooms(rquest，stay，roomcount)

验证入住不得超过stay天，一次预定房间不能超过roomCount

4、三级子类

crawler.supplier.base.HotelAvailabilityWithPaxRequestRewriter

覆盖validateRoom

--getRoomType

--roomKey

覆盖roomKey

roomKey = room.getAdults() + **"|"** + room.getChildren() + **"|"** + ages

与父类方法相比少了roomType的拼接。

覆盖getRoomType

--roomKey

覆盖roomCombineKey(room,index)

返回index+””

5、四级子类

crawler.supplier.base.HotelAvailabilityWithMaxPaxRequestRewriter

覆盖rewriteRooms 可用来被覆盖

实现将儿童转为成人，并设置每个房间人数相同，值为所有房间中最多的人数。

并调用了父类中的rewriteRooms

例：

List<Room> Room(2,1) Room(2) Room(1)

🡺 List<Room> Room(3) Room(3) Room(3)

父类中rewriteRooms功能：

因为继承层级中并没有覆盖rewriteSimilarRoom，所以房型组合没有作用，只是调用了simplifyRoomList根据上级父类的roomCombineKey进行了组合。

List<Room> Room(3) Room(3) Room(3)

🡺 List<Room> Room(3)

--rooms=3

--roomIndex=1,2,3

新方法toGdsGuest 可用来被覆盖

将儿童全部转为成人，年龄简单合并。

更好的实现方式：

对已知的Rewrite进行小功能拆分，内置到父类的rewriteRooms内部，默认基础功能处理，如果需要其他修改，则对个模块功能覆盖重新实现即可。当前模块拆分不明显，注释不清晰，往往需要覆盖顶层rewriteRooms方法。

应当实现为，父类实现总方法，子类不断抽象出功能层，只需要覆盖特有功能即可。或者对常用功能封装为单独的Rewrite，直接继承即可（已实现一部分）。还需要继续进行整理才能让逻辑更清晰。