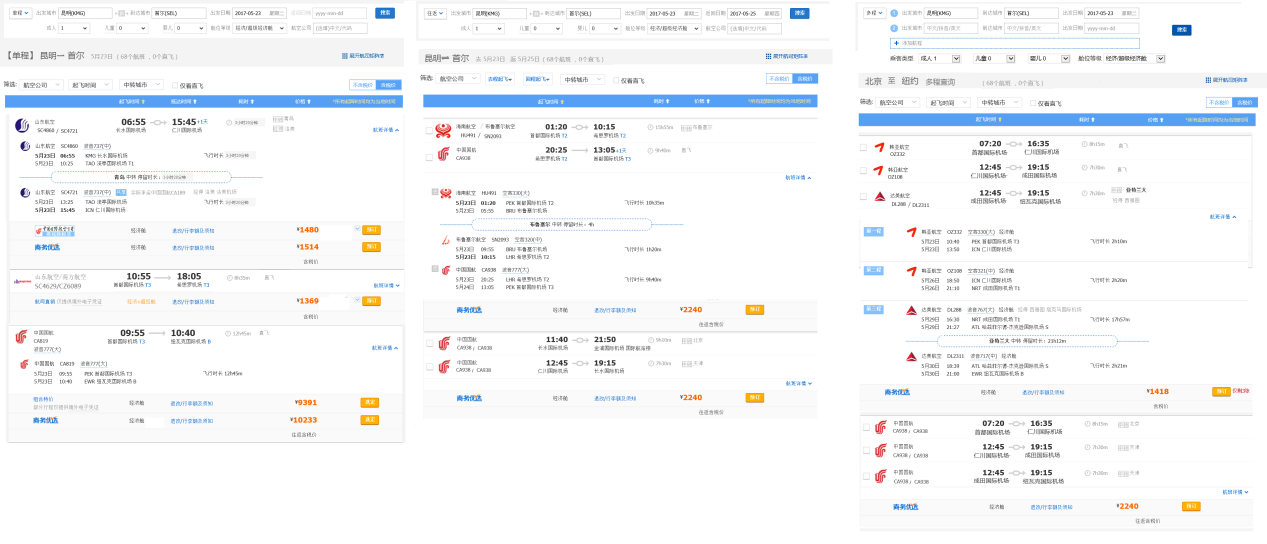
# 单程、往返、多程查询结果图示



航班结果列表

工具条

搜索输入框

为了方便对比将 单程、往返、多程的查询结果页放在一起。

查询结果页由三部分构成：搜索输入框、工具条、航变结果列表

# 搜索输入框

搜索输入框的动态效果可参考携程（[www.ctrip.com](http://www.ctrip.com)）。

### 单程和往返





单程、往返的唯一差别是 单程的“返回日期”项是disable的。

### 多程



多程跟单程、往返的差别比较大。 点“添加航程”会出来一个新的航段条件（红色圈起来的部分）默认放两个航段条件即可。

### 各输入项的说明

出发城市、到达城市、出发日期、返回日期、航空公司 这几个输入框 需要加控件。 控件用51book现有控件即可。

另外，希望几个关键的输入控件可以在当前控件输入完成后自动跳到下一个。（效果可参携程，今日天下通的响应更快，感觉更好 www.jinri.net）

单程的顺序是： 出发城市、到达城市、出发日期

往返的顺序：出发城市、到达城市、出发日期、返回日期

多程的顺序：逐个航段的 出发城市、到达城市、出发日期，且下一航段的出发城市默认上一航段的到达城市。

**各输入框需加校验：**

单程：出发、到达、出发日期，为必填项。

往返：出发、到达、出发日期、到达日期，为必填项。

往返：各航程的 出发、到达、出发日期 均为必填项（添加出来的航程也不例外）。

出发城市、到达城市， 这两个输入框 最终传給后台的是三字码（表示机场或城市的三字代码，例：BJS）。无效的三字码不允许提交表单。

出发日期、返回日期，均是日期格式。

单程、往返、多程： 所有情况下出发日期不得早于当天。

往返：出发日期不得早于当天。返回日期不得早于出发日期。

多程：后一程的出发日期不得早于前一程的出发日期。

希望通过日期控件做限制日期的选择

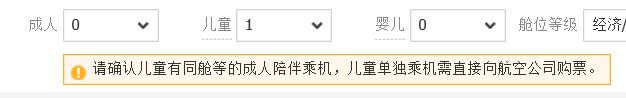
**乘客人数的校验：**

成人的下拉框可选值为（0-9）。儿童、婴儿下拉框可选值为（0-9）

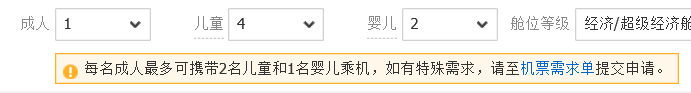
1、当婴儿、儿童、成人 人数都是0时。 提示，并禁用“搜索”按钮



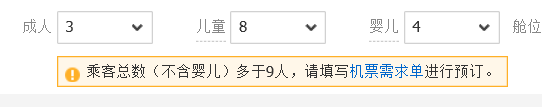
2、当（成人=0）且（婴儿+儿童）>0时。提示，并禁用“搜索”按钮



3、当 （儿童>成人\*2） 或者 （婴儿>成人）。提示，并禁用“搜索”按钮



4、当（成人+儿童+婴儿）>9。 提示，并禁用“搜索”按钮



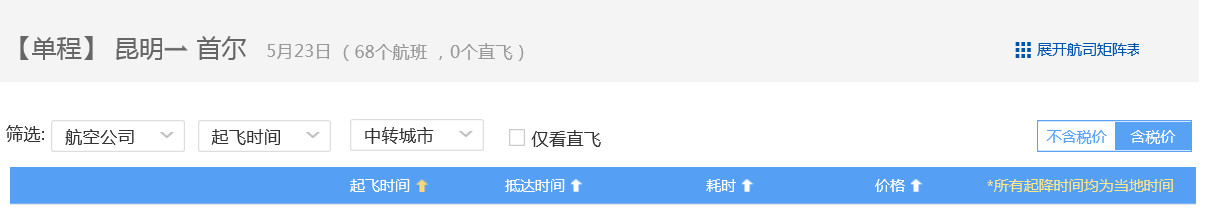
以上四个校验的提示文本回头给出，文本里含超链接。

（以上所有校验的效果均是用户完成输入后即时校验，效果可参看携程）

# 工具条

单程

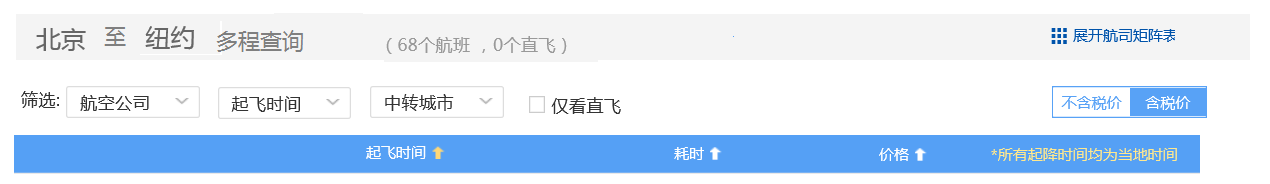
Title



往返



多程



注意各种行程类别下，这部分的区别：

在Title部分，单程、往返、多程显示各不相同。 （已经考虑过实现的可行性，就按图示显示）

筛选部分：往返程 多了“去程起飞时间”和“回程起飞时间”的筛选。 单程和往返只有“起飞时间”筛选，是指第一程的起飞时间。

排序部分：单程多了“抵达时间”的排序。 往返和多程只有“起飞时间”“耗时”“价格”的排序。

在往返和多程上，“耗时”的排序是指行程的总耗时。 “价格”的排序指某个航班方案的最低价排序（一个航班方案可能有多个可用价格）。

 点击这个图标，会展开如下的航班矩阵：



列出所有航司及不同中转次数下的价格。（对应返回json串中的priceTalbe）。 点击其中某个航司的列可以对搜索结果做按此航司筛选。

# 航班结果列表

### 一个航班方案

查询结果由多个航班方案组成。

返回的json串中有一个 avFlightList，avFlightList中的每个成员表示一个完整的航班方案。

每个航班方案由“航程信息”“航班详情”“可用价格列表”三部分构成。

“航程信息” 对应 avFlight.odList, “航班详情”由所有od的flightDetail构成，“可用价格列表”对应avFlight.fareList。

航程信息

下图是单程查询结果中一个中转航班方案的案例。



可用价格列表

航班详情

### 航程信息：

航程信息由 1到n个OD信息构成。（O——出发地，D——抵达地）

单程由1个OD构成。

往返由2个OD构成，一去一回。

多程由多个OD构成，取决于查询时传入了几段航程。

单程：



往返：



**② ⑨ ⑩ ⑾ ⑿**

1. **③ ④ ⑤ ⑥ ⑦ ⑧**

多程（案例是三段OD，最少2段、最多6段）：



不管是单程、往返、多程， 展示航程信息时， 只要将avFlight.odList 里的所有od遍历做展示即可。

每个od由上图红框中的 10个信息构成：

1. 航司：对应 od.airline
2. 航班号：od.flightNo
3. 出发时间：od.flightDetail[0].departureTime 此OD第一个航班的出发时间
4. 直飞中转标志：直飞标志是 中转标志是 
5. 到达时间：od.flightDetail[flightDetail.lenght-1].arriveTime 此OD最后一个航班的到达时间
6. 到达时间是否跨日期：

od.flightDetail[0].departureTime的日期 和od.flightDetail[flightDetail.lenght-1].arriveTime的日期 之间跨越的天数。第一个航班的起飞日期和最后一个航班的起飞日期跨越的天数。

比较便捷的计算方法是(Math.floor(D1/86400000)-Math.floor(D2/86400000))

D1=od.flightDetail[flightDetail.lenght-1].arriveTime

D2=od.flightDetail[0].departureTime

1. 航程时长：od.duration
2. 经停or中转信息：
   1. 如果od.stopCity 和od.transferCity 都是空，显示“直飞”。
   2. 如果 od.transferCity不为空，显示 ，中转图标后面的“布鲁塞尔”取od.transferCity。
   3. 如果 od.stopCity不为空，显示，经停图标后面的“西雅图”取od.stopCity
3. 出发机场：od.dPortName
4. 出发机场航站楼：如果od.flightDetail[0].depTerm 不为空，若此字段不为空，若为数字，显示 “T”+此值，若为字符,直接显示此值。为空不显示
5. 到达机场:od.aPortName
6. 到达机场航站楼：od.flightDetail[flightDetail.length-1].arrTerm不为空，若此字段不为空，若为数字，显示 “T”+此值，若为字符,直接显示此值。为空不显示

### 航班详情

单程的 中转 案例



单程的直飞案例



往返的航班详情案例



多程的航班详情案例：



单程的每个航班方案只有OD

往返的每个航班方案由两个OD构成，第一个OD前显示 ，第二个OD前显示 

多程的每个航班方案由多个od（2~6个）构成，每个od的航班前显示图标 或者 第二程、第三程 ……

每个OD都有 flightDetail，是个list，包含一到多个成员。

如果flightDetail有多个成员，表示这个OD由多个航班构成，各航班间用连接。

以下面这个多程的查询结果为例：

OD1

OD2

OD3

OD1.flightDetail

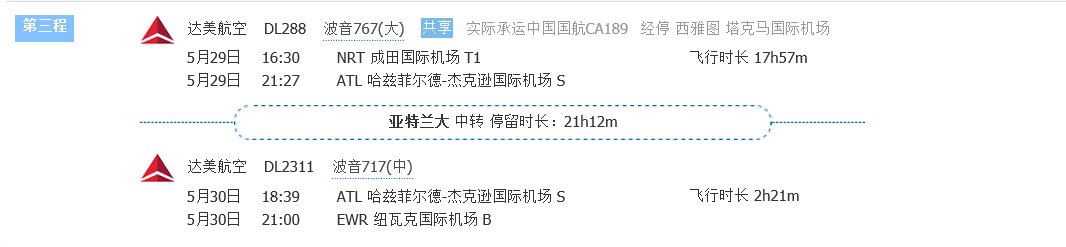
OD2.flightDetail

OD3.flightDetail



航程信息里的每个OD，对应到 下方“第一程”“第二程”“第三程”的航班详情。

其中“第一程”“第二程”都只由一个航班构成，“第三程”由2个航班构成。



1. **② ③ ④**

**⑤ ⑥ ⑦ ⑧ ⑨ ⑩**

**⑾ ⑿ ⒀ ⒁ ⒂**

**⒃ ⒄**

用flightDetail[0]代表 od.flightDetail这个list中的第一个成员

1. 航司：flightDetail[0].airlineName
2. 航班号：flightDetail[0].airline + flightDetail[0].airFlightNo
3. 机型：flightDetail[0].plane
4. 共享航班和经停信息：
   1. flightDetail[0].codeshare 如果是 true，表示此航班是共享航班，显示：+“实际承运”+ flightDetail[0].airCarrierName + flightDetail[0].airCarrierAline + flightDetail[0].airCarrierFlight
   2. flightDetail[0].stopInfoList 这是个list，如果这个list不为空。遍历list，显示 “经停”+cityName+airportName。cityName表示经停点城市名，airportName表示经停点机场名。
5. 此航班的出发日期：flightDetail[0].departureTme 转化为日期格式后 取 月、日，MM-dd
6. 此航班的出发时间：flightDetail[0].departureTme 转化为日期后去 时、分，hh-mm
7. 出发机场三字码：flightDetail[0].departurAirport
8. 出发机场名称: flightDetail[0].dPortName
9. 出发机场航站楼：flightDetail[0].depTerm，若此字段不为空，若数字，显示 “T”+此值，若为字符,直接显示此值。为空不显示
10. 此航班的飞行时长：flightDetail[0].duration
11. 此航班的抵达日期：flightDetail[0].arriveTIme转化为日期格式后 取 月、日，MM-dd
12. 此航班的抵达时间：flightDetail[0].arriveTme 转化为日期后取 时、分，hh-mm
13. 抵达机场三字码：flightDetail[0].arriveAirport
14. 抵达机场名称: flightDetail[0].aPortName
15. 抵达机场航站楼：flightDetail[0].arrTerm，若此字段不为空，若数字，显示 “T”+此值，若为字符,直接显示此值。为空不显示
16. 中转点：连接前后两个航班



技术上的处理方式是 在处理完当前这航班之后，如果后续还有航班，取当前航班的抵达城市 flightDetail[index].aCityName

和当前航班的抵达时间和下一航班的出发时间的时间差。timeDiff(flightDetail[index+1].departureTime, flightDetail[index].arriveTime)

**function** timeDiff(d1,d2){

**var** mm= ((d1-d2)/60000)%60 ;

**var** hh=Math.floor((d1-d2)/3600000);

**return** hh+'小时'+mm+'分钟'

} }

### 可用运价列表

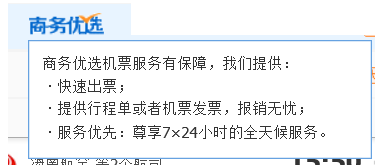
1. **② ③ ④ ⑤ ⑥**



表示这个航班的可预订的价格有哪几个。每个avFlight都有一个 fareList，每个成员表示一个可预订价格。

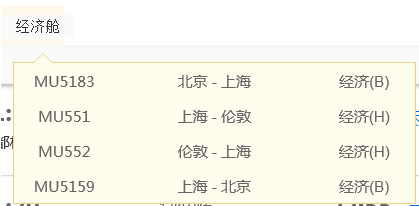
这部分有比较多的tips。

**1、运价的title**，会存在几种不同定义的运价。不同的Title有不同的tips内容。



**2、服务等级信息：fareList[i].physicalClassName**

**Tips里的内容取fareList[i].classInfo 是一个list，遍历。**



**3、退改及行李额须知。**

这个内容中的一部分信息需要再向服务器端发一次请求，请求内容的要从avFlight里取数据。在数据提交章节详述。



**4、价格：**

价格这里有几种不同的情况。

如果工具条上的选项是“往返含税价” 

则价格这里显示 **fareList[i].avgSettlementPriceWithTax**

**且当乘客类型是几种乘客类型混合查询时，需要显示 tips**

**由于在搜索输入时对不同乘客类型的人数做了限制。**

**可查询的只有这几种情况：**

**a、成人：**



B、成人+儿童+婴儿 （人数按实际情况，下图仅为示例）

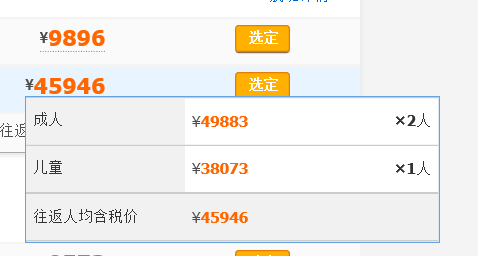
Tips里的信息分别取自**fareList[i].**adultPrice、**fareList[i].**childPrice、**fareList[i].**infPrice

价格= settlementPriceWithTax

人数=passengerCount



C、成人+儿童



D、成人+婴儿



另一种情况是工具条上选择了“不含税价格”

则价格处显示如下：



价格处是 **fareList[i].avgSettlementPrice ,**

**“税费”是fareList[i].avgTax**

Tips里显示如下（只给出了一种混合查询的案例，其他案例类推）：

Tips里的信息分别取自**fareList[i].**adultPrice、**fareList[i].**childPrice、**fareList[i].**infPrice

价格=settlementPrice

税=tax

人数=passengerCount

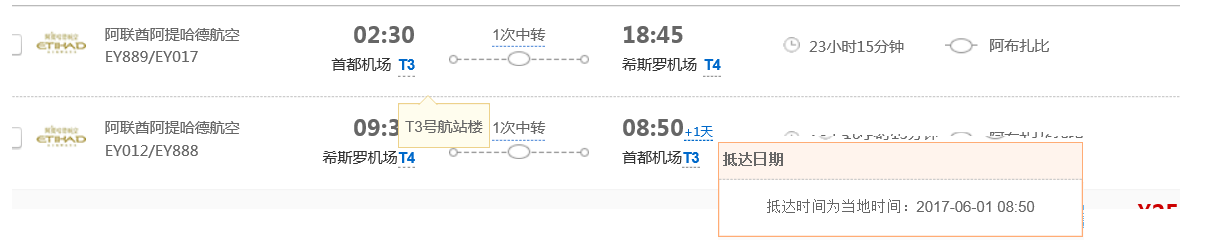


5、预订按钮：这里需要提交数据。具体参数在“数据提交”章节详述

6、座位数

当**fareList[i].seats <9时， 显示 :“仅剩”+ fareList[i].seats+”张”**

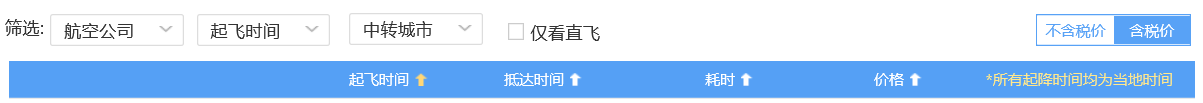
# 其他tips

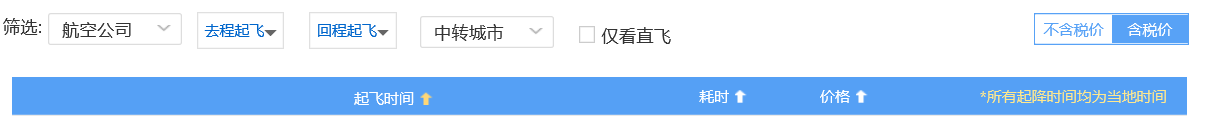


航程信息上的 航站楼 和 抵达日期的跨天数， 也需要加tips。

未来，在航班详情的 机型信息上，需要加tips。

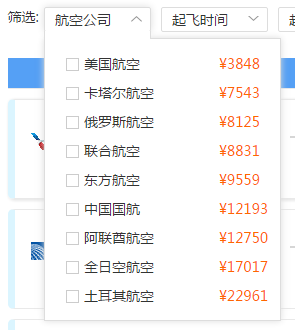
# 工具条的处理





### 筛选

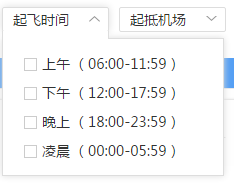
1. 航司筛选



允许多选。可选的航司通过遍历 priceTable得到。航司名取priceTable[i]. airlineName，价格取lp、lpo、lpt 中最低的那个。

当选中了某个航司时，航班结果列表中的 avFlightList[index].airlineSet包含了这个航司的留下。 用String的indexOf() 这个方法处理就可以，priceTable[x].airlineName 和avFlightList[index].airlineSet 这两个字段的内容都是预先处理过的，格式一致。

1. 起飞时间



起飞时间的可选内容是固定值。允许多选。

单程和多程，只出现一个“起飞时间”的筛选。

往返程提供 “去程起飞时间”“回程起飞时间”两个筛选。

avFlightList[i].departHour 表示第一程的起飞时间所在的小时。

avFlightList[i].rtDepartHour 仅在往返查询时有值，表示回程起飞时间所在的小时。

“起飞时间”或者“去程起飞时间”的筛选 是比较avFlightList[i].departHour

上午：6≤avFlightList[i].departHour <12

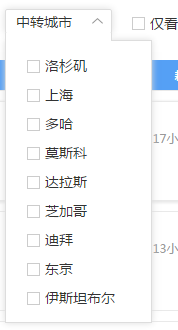
下午：12≤avFlightList[i].departHour <18

晚上：18≤avFlightList[i].departHour <24

凌晨：0≤avFlightList[i].departHour <6

“回程起飞时间”的筛选是 比较 avFlightList[i].rtDepartHour

1. 中转城市



“中转城市”筛选框的可选项取自json串中的 transferCity这个list。有几个列几个。

允许多选。

用户选中某中转城市时，avFlightList[i].transferCitySet这个字符串中含当前所选筛选城市的留下。

1. 仅看直飞

当用户勾选此项时。 avFlightList[i].transfer 等于0的留下。

1. 含税价/不含税价

在“可用运价列表”部分对此有说明了。

### 排序

1. 起飞时间

根据avFlightList中每个成员的odList中的第一个od的departureTime排序，也就是第一程的起飞时间

1. 抵达时间

只有单程查询有 抵达时间 的排序功能。 往返和多程没有这个排序功能

根据avFlightList中每个成员的odList中的第一个od的arriveTime排序，

1. 耗时

取avFlightList[i].totalDuration 排序

1. 价格

取avFlightList[i].fareList 中的第一个运价（也是价格最低的那个运价）进行排序。

# 数据提交

### 查询框提交搜索：

{

"journeyType": "RT", /\*行程类型,往返：RT，单程:OW ,多程：MS\*/

"segmentList": [ /\*航程列表， 单程是一个，往返是两个，多程几程几个\*/

{

"oriCode": "BJS",

"desCode": "NGO",

"departureDate": 1495036800000

}

],

"cabinClass": "ECONOMY;PREMIUM\_ECONOMY", /\*舱位等级\*/

"passengerType": [ /\*乘客类型\*/

{

"passgerType": "ADT", /\*成人人数\*/

"passgerNumber": 2

},

{

"passgerType": "CHD",/\*儿童人数\*/

"passgerNumber": 1

},

{

"passgerType": "INF",/\*婴儿人数\*/

"passgerNumber": 1

}

],

"directFlightsOnly": false, /\*是否仅查直飞,默认是false\*/

"airline": null,/\*航空公司，传航司二字码\*/

"debug": true

}

查询框数据提交

1. 单程

"segmentList": [ /\*航程列表， 单程是一个\*/

{

"oriCode": "BJS", 

"desCode": "NGO", 

"departureDate": 1495036800000 

}

],

1. 往返

"segmentList": [ /\*航程列表，往返是两个 \*/

{

"oriCode": "BJS",

"desCode": "NGO", 

"departureDate": 1495036800000

},

{

"oriCode": " NGO ",/\*取输入框里的到达城市作为回程起点\*/

"desCode": " BJS ",/\*取输入框里的出发城市作为回程终点\*/

"departureDate": 1495900800000

}

],

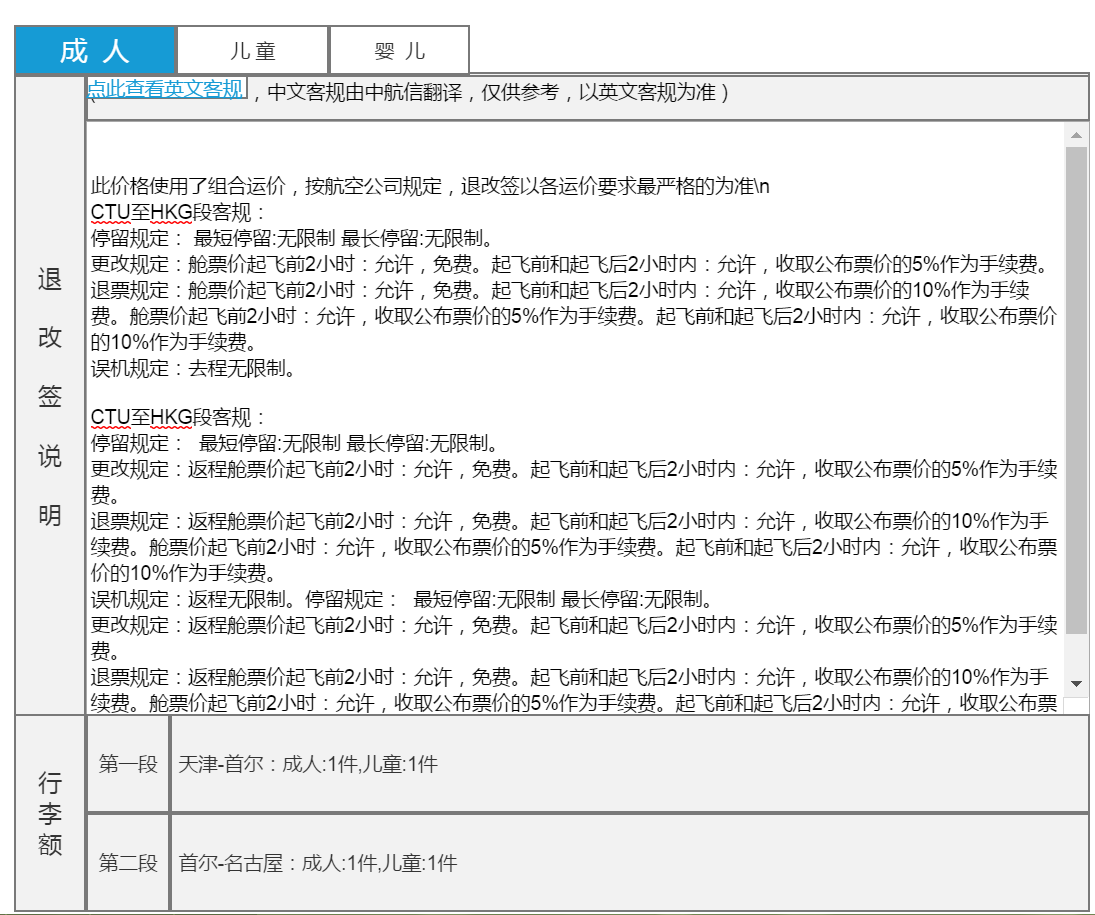
1. 多程：将输入框里的各航程顺次放入segmentList。oriCode放出发城市，desCode放到达城市

### 客规查看（这个比较复杂，文档看不明白就电话沟通）



运价列表的“退改签/行李额须知”，这里需要到后台查出退改签数据，再结合行李额进行展示。

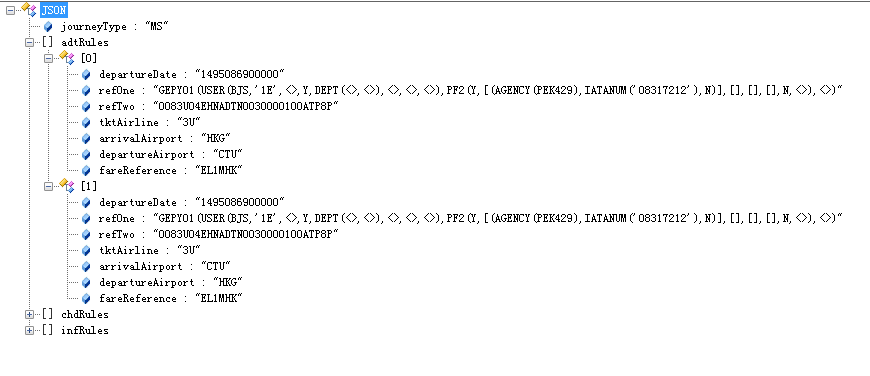
鼠标移到“退改/行李额说明”，显示下面的tips。



红色框里的文字可能很长，需要使用下拉条

这个tips里的退改签说明的内容通过请求服务器端的数据获得。行李额通过farelist[index].baggagesRemark 组装

下图是请求退改签规则的json的示例：



对应的后端java对象的定义如下：

**public** **class** TktRulesRQ {

**private** String journeyType;

**private** List<Tktrules> adtRules; //成人的退改签规定的查询条件

**private** List<Tktrules> chdRules; //儿童的退改期规定的查询条件

**private** List<Tktrules> infRules; //婴儿的退改期规定的查询条件

}

\*/

**public** **class** Tktrules {

**private** String departureDate;

**private** String refOne;

**private** String refTwo;

**private** String tktAirline;

**private** String arrivalAirport;

**private** String departureAirport;

**private** String fareReference;

**}**

这个json串的构建方式：

（业务知识：航空公司销售机票时，不同的价格会对应不同的退改签和行李额要求。 所以用于查询退改签说明的参数，在avFlightList[i].fareList[j]的 adultPrice、childPrice、infPrice 对象中。）

用户查看当前运价的退改签时，取 adultPrice.tktRules 赋值给TktRulesRQ.adtRules, childPrice. tktRules赋值给TktRulesRQ. chdRules， infPrice.tktRules 赋值给TktRulesRQ. infRules。

TktRulesRQ. journeyType，从当前的航班方案avFlightList[i].journeyType取值。

后端返回的json串的结构

{

"adtRule": [

"此价格使用了组……", //成人退改签的中文说明

"此价格使用了组……" //成人退改签的英文说明

],

"chdRule": [

"此价格使用了组……", //儿童退改签的中文说明

"此价格使用了组……" //儿童退改签的英文说明

],

"infRule": [

"此价格使用了组……", //婴儿 中文

"此价格使用了组……" //婴儿 英文

]

}

将成人、儿童、婴儿的退改签说明分布显示在tips的对应页签中。 如果没有对应乘客的退改签信息，则不显示。 如果 只有一种乘客的退改签， 则不显示页签头。

每种乘客类型的 中英文退改签说明可以切换展示。

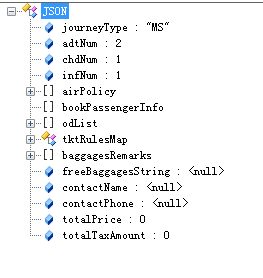
### 预订按钮提交



当用户预订按钮时，需向后端提交数据，并切换到输入乘客信息的页面。

当用户点击所选的航班方案下的某个运价的预订按钮时， 需先找到用户选定的是哪个航班方案，以及这个航班方案下的哪个运价。 假设用户选了 avFlightList[i] 这个航班方案，使用了 avFlightList[i].fareList[0]这个运价：

向后端提交的booking请求的结构如下。



需要赋值的是如下字段：

1. journeyType：取avFlightList[i].journeyType
2. airPolicy:

airPolicy是个policyInfo构成的list，取

avFlightList[i].fareList[0].adultPrice.policyInfo

avFlightList[i].fareList[0].childPrice.policyInfo

avFlightList[i].fareList[0].infPrice.policyInfo 放进这个list。

1. odList:

取avFlightList[i].odList赋值给它

1. baggagesRemarks:

取avFlightList[i].fareList[0].baggagesRemark 赋值给它

1. airPolicy 、tktRulesMap、adtNum、chdNum、infNum:

赋值伪代码：

**if** (avFlightList[i].fareList[0].adultPrice!= **null**) { //处理成人的信息

bookingJson.airPolicy.add(avFlightList[i].fareList[0]. adultPrice.policyInfo); //成人对应的价格信息

bookingJson.tktRulesMap.put(avFlightList[i].fareList[0]. adultPrice.passengerType, avFlightList[i].fareList[0]. adultPrice.tktRules); //成人对应的客规

bookingJson.adtNum= avFlightList[i].fareList[0]. adultPrice.passengerCount); //成人人数

}

if (avFlightList[i].fareList[0].childPrice!= null) { //处理儿童的信息

bookingJson.airPolicy.add(avFlightList[i].fareList[0]. childPrice.policyInfo);

bookingJson.tktRulesMap.put(avFlightList[i].fareList[0]. childPrice.passengerType, avFlightList[i].fareList[0]. childPrice.tktRules);

bookingJson.chdNum= avFlightList[i].fareList[0]. childPrice.passengerCount);

}

if (avFlightList[i].fareList[0].infPrice!= null) { //处理婴儿的信息

bookingJson.airPolicy.add(avFlightList[i].fareList[0]. infPrice.policyInfo);

bookingJson.tktRulesMap.put(avFlightList[i].fareList[0]. infPrice.passengerType, avFlightList[i].fareList[0]. infPrice.tktRules);

bookingJson.infNum= avFlightList[i].fareList[0]. infPrice.passengerCount);

}

airPolicy 是个由 policyInfo构成的list；tktRulesMap是个有乘客类型和乘客类型对于的退改签构成的Map。

}