RD

中国民航信息网络股份有限公司

运价项目

模块设计说明书

版 本 号：

编制部门：　　中国民航信息网络股份有限公司研发中心分销产品研发部

编制人：　　编制日期：

审批人：　　审批日期：

变 更 记 录

序号	修改原因	修改目的	修改内容	修改人/日期	审批人/日期	修改后的版本号	实施日期
1							
2							
3							
4							
5							
6							
7							
8							
9							
10							
11							
12							
13							

目录

1.术语定义

1、pricing：按航段查询运价，指用户根据航段查询运价，得到运价和相关政策的功能。包括国际 pricing 和国内 pricing 两个模块。

2、Pricing 接口：只用户根据航段请求本系统，经过本系统的处理返回给用户请求航段得到的运价和相关政策的接口。包括国际 pricing 和国内 pricing 两个模块，不同的模块内部实现、调用的底层接口各不相同。

3、国际 pricing 接口：pricing 接口的国际部分。

4、国内 pricing 接口：pricing 接口的国内部分。

5、总则：一个用户可以为每个航空公司设置一个总则。因为大多数航空公司的总则可能是一样的，为了方便用户，将设置通用总则。

6、政策：指航空公司于代理商、代理商与代理商之间的一些关于票价和返点的一些约定，用于机票分销的各级代理商之间，可用作政策匹配。

7、国际政策导入:是指用户将指定格式的 excel 国际政策数据批量导入到数据库中。

8、ODD 点 ：指的是去程起点（O）、去程终点或者折返点（D）、回程终点（D）。

9、一代：

10、二代：

11、政策匹配：

2.前提说明

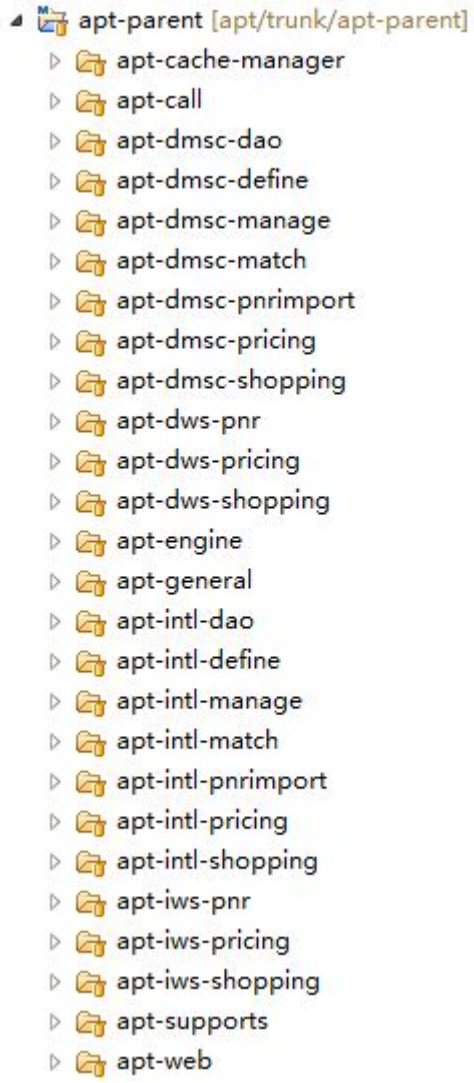
2.1 环境说明

环境	软硬件环境		版本/套餐
开发环境	模块设计工具	Visio	Microsoft Office Visio 2007
	开发工具	Eclipse	Integration V3.7 indigo
	文档编写	Word	Microsoft Office Word 2007

测试环境	单元测试工具	JUnit	JUnit Testing Framework V4

2.2 模块命名规则

模块图、以及每个模块所完成的相关功能。



3.详细设计

3.1 国际政策维护模块详细设计

3.1.1 模块概述

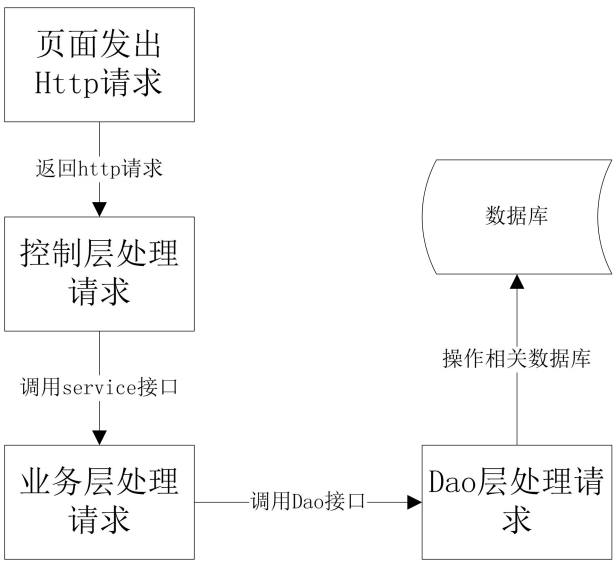
国际政策维护模块，是指用户可以通过模块随时增加政策、删除政策、修改政策、查询政策，达到对政策的日常维护。

3.1.2 模块的需求描述

国际政策维护模块，用户在需要增加、删除、修改、查询政策的时候，可以实现相应的操作政策。每个用户都只能操作自己的政策。

3.1.3 模块的设计思路

根据需求，我们得出如下设计思路图：



简单的说就是用户通过操作我们的系统,如添加政策时,录入政策信息，保存到我们系统的数据库信息中；查询时，即从数据库中取出信息，显示在系统界面上；删除时，我们并不

是真正的删除政策信息，而是通过设置截止时间，假定删除一条政策，把数据依然存在数据库中，以备查阅。

3.1.4 模块的数据库设计

根据需求我们设计出一张数据库表结构，国际政策表，如下图：

表 1 国际政策信息表

列名	数据类型	默认值	是否主键	描述
id	Object		是	政策 UUID
tktAirline	String		否	出票航司
distributor	String		否	投放分销商
wholesaler	String		否	批发商
passagerType	String		否	旅客类型
passagerCount	String		否	旅客人数
groupType	String		否	团散类型
travelType	String		否	行程类型
outOrg	String		否	去程起点
outOrgEx	String		否	去程起点除外
outDes	String		否	去程终点
outDesEx	String		否	去程终点除外
returnDes	String		否	回程终点
ReturnDesEx	String		否	回程终点除外
firstTravelDate	String		否	最后一国际段旅行日期
lastTravelDate	String		否	最后一国际段旅行日期
startTktDate	String		否	出票日期开始时间
endTktDate	String		否	出票日期结束

isCarrierOne	Boolean		否	是否同意承运人
isNoAlloewShareCode	Boolean		否	不允许代码共享
allowShareCode	String		否	允许和？？代码共享
isDirectFlight	Boolean		否	是否必须直飞
noAllowPass	String		否	不允许经过
mustPass	String		否	必须经过
allowFlightNo	String		否	仅限航班号
excludeFlightNo	String		否	排除航班号
cabinOne	String		否	舱位一
rebateOne	Float		否	返点一
cabinTwo	String		否	舱位二
rebateTwo	Float		否	返点二
cabinThree	String		否	舱位三
rebateThree	Float		否	返点三
mixCabin	String		否	混仓返点选择
poundage	Integer		否	混仓手续费
proxyFree	Integer		否	代理费
sellConfig	String		否	销售配置
remark	String		否	备注
Status	String		否	状态
workDate	String		否	生效日期
endDate	String		否	生效截止日期
tktAirlines	String[]		否	数组字段出票航司

distributors	String[]		否	数组字段分销商集合
passagerTypes	String[]		否	数组字段旅客类型
groupTypes	String[]		否	数组字段团散类型
outOrgs	String[]		否	数组字段去程起点
outOrgExs	String[]		否	数组字段去程起点除外
outDess	String[]		否	数组字段去程终点/折返点
outDesExs	String[]		否	数组字段去程终点除外点
returnDess	String[]		否	数组字段回程终点
returnDesExs	String[]		否	数组字段回程终点除外
firstTravelDates	String[]		否	数组字段第一国际旅行日期
lastTravelDates	String[]		否	数组字段最后一国际段旅行日期
allowShareCodes	String[]		否	数组字段允许共享航班号
noAllowPasss	String[]		否	数组字段不允许经过点
mustPass	String[]		否	数组字段必须经过点

allowFlightNos	String[]		否	数组字段允许航班号
excludeFlightNos	String[]		否	数组字段排除航班号
cabinOnes	String[]		否	数组字段舱位一
cabinTwo	String[]		否	数组字段舱位二
cabinThree	String[]		否	数组字段舱位三
travelTypes	String[]		否	数组字段行程类型
importKey	String		否	本次导出编号（用于政策导入）
userId	Long		否	用户 ID

3.1.5 模块的限制条件

对于政策维护模块,分别有两个有效性验证，一是政策添加和修改页面的页面限制条件，我们必须限制有些必输项以及部分数据需要校验格式的正确性，具体情况如下表：

表 2 国际政策信息表

名称	类别	规格	备注
基础信息			
出票航空公司	文本框，必填	两字码，多个用斜线分隔，小写自动转大写。	举例：AF/KL 需要动态校验两字码是否存在
投放分销商	文本框，必填	自由文本，多个用斜线分隔	不校验，匹配时和传入内容比对
批发商	文本框，	自由文本	不校验
旅客信息			

旅客身份	文本框，必填	填写：成人，学生，劳务，海员，探亲，移民，外交官，军人，多个用斜线分隔。默认成人。	校验：
旅客人数下限	文本框，必填	数字（3），默认 1	
政策适用类型	文本框，必填	填写：散客，团队，多个用斜线分隔。	举例：散客/团队
地理位置信息			
行程类型	下拉框	选项：OW、RT、OW/RT，默认 OW/RT	
去程起点	文本框，必填	机场/城市三字码或中文名，多个用斜线分隔。	举例:欧洲/北美洲/中国大陆/港澳台/FRA/MIL; 基础数据，需要校验是否存在
去程起点除外点	文本框，	机场/城市三字码或中文名，多个用斜线分隔。	基础数据，需要校验是否存在
去程终点（折返点）	文本框，必填	机场/城市三字码或中文名，多个用斜线分隔。	基础数据，需要校验是否存在
去程终点（折返点）除外点	文本框，	机场/城市三字码或中文名，多个用斜线分隔。	基础数据，需要校验是否存在
回程终点	文本框	机场/城市三字码或中文名，多个用斜线分隔。	基础数据，需要校验是否存在
回程终点除外点	文本框	机场/城市三字码或中文名，多个用斜线分隔。	基础数据，需要校验是否存在
日期要求			
第一国际段旅行日期	文本框 必填	yyyy-mm-dd，范围用~，多个斜线分隔。	举例： 20140101-2014-9-30/2014-10-8~2014-12-31
最后一国际段旅行日期	文本框	yyyy-mm-dd，范围用~，多个斜线分隔。	举例： 2014-12-25~2014-12-31/2015-1-1
出票日期范围	文本框 必填	yyyy-mm-dd，范围用~，	举例： 2014-1-1~2014-12-31
更多要求			
全程同一承运人	勾选框	默认为空，即不选择	
不允许代码共享	勾选框	默认为空，即不选择	
仅允许和代码共享	文本框	航空公司两字码，多个用斜线分隔。	举例： CZ
必须直飞	勾选框	默认为空，即不选择	
不能经过	文本框	机场/城市三字码，多个用斜线分隔。	举例： LON/PAR
必须经过	文本框	机场/城市三字码，多个用斜	

		线分隔。	
排除航班号		两字码+航班号。 可以是单个航班或范围，*号表示一位任意数字，需要填写航空公司两字码，多个用斜线分隔。	
仅限航班号			
政策舱位+返点			
舱位	文本框 必填	填写舱位字母，可以多个，斜线分隔	举例：Y/H/B;
返点	文本框， 必填	正数（一位小数），默认 0	举例：2.5；（表示 2.5%）
舱位	文本框	填写舱位字母，可以多个，斜线分隔	举例：Y/H/B;
返点	文本框	正数（一位小数），默认 0	举例：2.5；（表示 2.5%）
舱位	文本框	填写舱位字母，可以多个，斜线分隔	举例：Y/H/B;
返点	文本框	正数（一位小数），默认 0	举例：2.5；（表示 2.5%）
混舱选择	下拉框	可以选择： 1/2 或 较低	
手续费	文本框	正或负整数，默认 0	举例：-10
代理费	文本框	数字（2），最多保留一位小数	区别于航空公司默认代理费时，可以填入。
销售配置	文本框	自由文本	不校验
销售备注	文本域	字符串（250）	
提交	按钮		点击提交时提交页面所有数据，并进行格式校验，校验通过后再提交后台，完成政策添加。

二是政策查询页面时,我们需要限制查询的条件,只能根据那些条件来查询政策，具体情况如下表：

表 3 国际政策信息表

名称	类型	规格	备注

出票航司两字码	文本框，必填	数字/字母（2）	航空公司两字码，查询部区分大小写
出票时间	文本框	日期控件	yyyy-MM-dd 默认是当前日期
去程起点	文本框	字符串	
去程终点	文本框	字符串	
回程终点	文本框	字符串	
生效日期	文本框	日期控件	yyyy-MM-dd 默认是当前日期

3.1.6 模块的代码设计

政策维护模块，外部用户主要是通过相关界面程序来操作我们系统的，用户只需要操作相应的按钮，即可操作我们系统。

在我们系统内部，主要是通过接受 web 页面传来的数据，然后通过 DB 连接操作数据库，从而达到对政策数据的基本维护。下图是我们这个模块涉及到的相关类图：

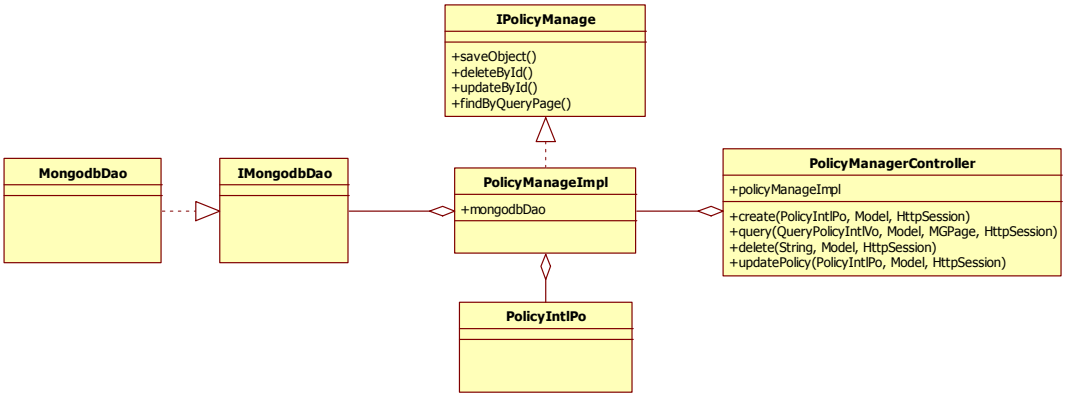


图 1 政策维护相关类图

在政策维护模块,我们主要功能实现是在 policiManageImpl 这个类里面，包括对政策的操作以及其他方法。在进行相关数据处理时,具体流程如下图：

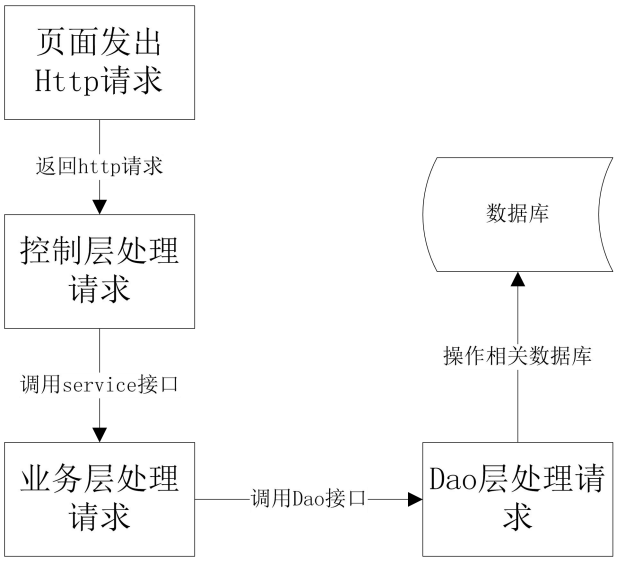


图 2 政策维护相关流程图

3.1.6.1 页面发出 Http 请求

这个步骤是在用户操作 web 页面的时候发出的，用户通过点击页面上的按钮或者其他操作方式,向我们系统发出添加、修改、删除、查询等请求。

3.1.6.2 控制层处理相关请求

在这个步骤中，控制层主要负责将提交的请求分配给每个处理请求的方法，如添加请求，我们将调用 create 方法来处理，依次类推，在处理方法中我们会对传输的数据做一些格式或者验证处理，但我们不会对其进行逻辑处理。

3.1.6.3 业务层相关处理

这个步骤主要是对每个请求的逻辑进行处理的步骤。用户想要添加一个政策,我们把经过格式处理的数据，写入一些逻辑，如相关时间，相关人等等，最后调用 Dao 层接口处理数据。

3.1.6.4Dao 层处理

Dao 层主要是数据访问层，是访问数据的接口层,这里我们可以通过一些框架或者直接用 Sql 语句来操作数据库，因此,我们收到从业务层传入的相关请求，然后按照方法对数据库进行相应的操作。

3.1.6.5 数据库

这里主要是存放我们政策数据信息的。

3.1.7 模块的异常情况

本模块的异常情况主要有：

1、用户必填参数没有填写或者填写不符合规范的情况下，会直接在 web 页面提示用户。

2、相关逻辑异常,如果发现有相关逻辑的异常,我们会终止当前操作，并撤销已经操作的操作，并提示用户。

3.1.8 模块尚未解决的问题

暂无

3.2 国际总则模块详细设计

3.2.1 模块概述

国际总则模块，基本信息

数据库中表名	tb_general
对应表字段的类	GeneralPo.java
使用的数据库	mysql
dao 层类	GeneralDao.java
service 层类	GeneralService.java
web 层类	GeneralController.java
增加总则页面	generalAdd.jsp
增加总则 js	add.js
修改总则页面	modifyGeneral.jsp
修改总则 js	add.js
查询总则页面	queryGeneral.jsp
查询总则 js	query.js
复制总则页面	copyGeneral.jsp
复制总则 js	add.js

3.2.2 模块的需求描述

一 无奖励情况

□ 婴儿无奖励
□ SOTO 票，无奖励。
□ IT 票，无奖励。
□ Open 票无奖励
□ 含_____票价基础无奖励
□ 票价低于____，无奖励

二 OD 设置

去程起点

☐ 出票航司的第一个航段的起点　（基础版默认配置）
☐ 出票航司的实际承运的第一个航段的起点
☐ 第一个国际段的起点

去程终点（折返点）

☐ 出票航司飞到的最远点（里程）　（基础版默认配置）
☐ 跨大区（或子区或国际）段的终点
☐ 出票航司飞到的最远点所在的运价单元的终点

回程终点

☐ 出票航司的最后一个航段的终点　（基础版默认配置）
☐ 出票航司实际承运的最后一个航段的终点
☐ 最后一个国际段的终点

三 同盟航空公司设置

组 1：_____　_____　_____　_____
组 2：_____　_____　_____　_____
组 3：_____　_____　_____　_____
组 4：_____　_____　_____　_____

四 数据选取方式

☐ 仅返回数据日期最新的

☐ 仅返回结算价最高的

☐ 仅返回结算价最低的

五 结算价计算方法

结算价方法 1 - 整体调整

☐ 非实际承运里程占比　区间 1__　返点调整_____；区间 2__　返点调整_____；
☐ Addon　1 段　返点调整_____；　2 段 返点调整_____；
☐ SPA　1 段　返点调整_____；　2 段 返点调整_____；

公式：结算价=票价*（1-代理费）*（1-返点）+ 税款 + 手续费

结算价方法 2 - 调整记入奖励的部分

计入奖励的航段是：
默认主航段计入奖励。

☐ Add On 计入奖励
　☐ 自营的 Addon 段计入奖励
　☐ 非自营的 Addon 段计入奖励

□ SPA 计入奖励

□ 自承运的 SPA 段计入奖励

□ 与主航段同一票价计算组计入奖励

□ 与主航段不同票价计算组计入奖励

□ 非自承运的 SPA 段计入奖励

□ 与主航段同一票价计算组计入奖励

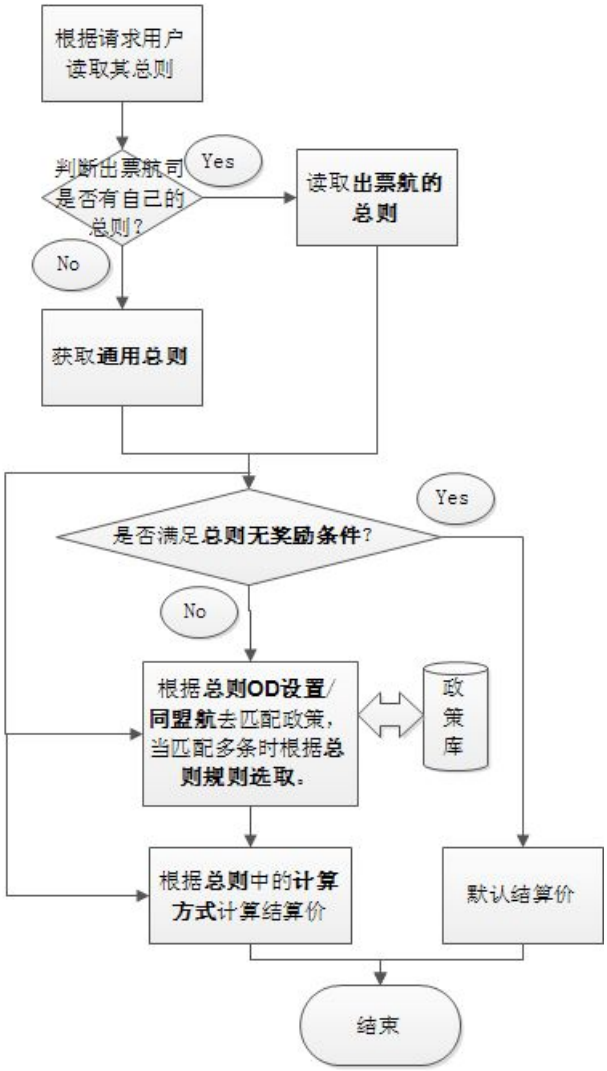
□ 与主航段不同票价计算组计入奖励

□ Q 值计入奖励

公式：结算价=（记入奖励之和）*（1-代理费-返点）+ （票价-记入奖励之和）*（1-代理费） + 税款 + 手续费

3.2.3 模块的设计思路

根据需求，我们得出如下设计思路图：



首先需要获取总则，一个用户可以为每个航空公司设置一个总则。因为大多数航空公司的总则可能是一样的，为了方便用户，将设置通用总则。

读取总则时，如果出票航有自己的总则，将读取其总则，如果没有的话，将读取通用总则。不同用户之间的总则数据是独立的。

3.2.4 模块的数据库设计

总则这一块用到的数据库是 mysql.

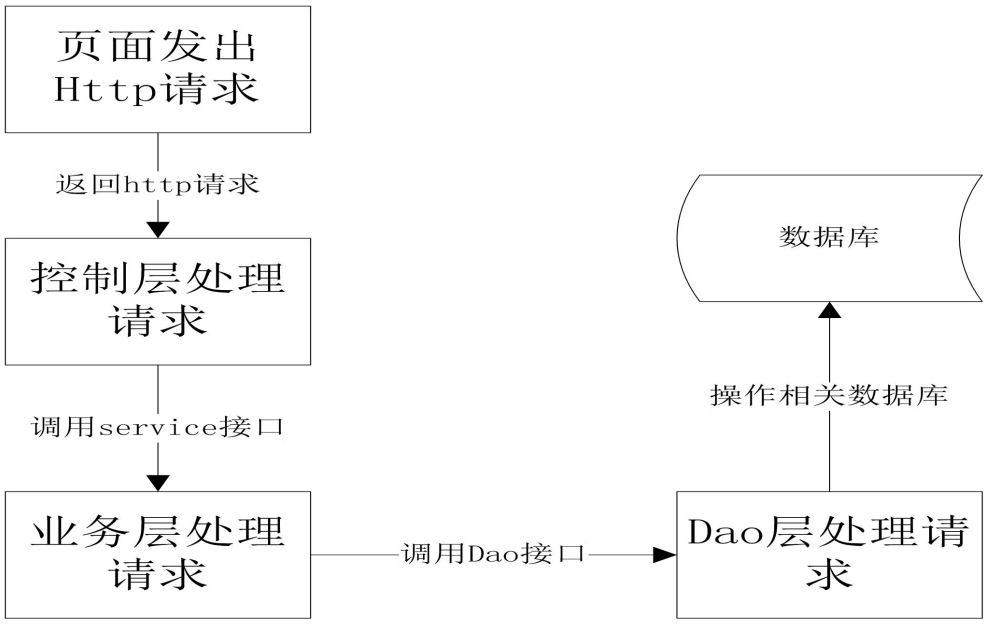
3.2.5 模块的限制条件

每个用户的总则不可重复，每个用户之间的总则是相互独立。每个用户都可为相应的航空公司设定唯一的一条总则，也可设置一条通用的总则，适用于大多数航空公司。

3.2.6 模块的代码设计

主要的功能是用户操作界面，对每条总则数据的增加 删除 修改 查询 复制。

然后传递参数，我们接收这些参数返回相应的结果到页面上，主要的实现流程如图：



几个功能的实现情况：

增加：增加这一块其中有一个后台的异步校验二字码是否存在，添加前，先用当前登录用户的 id (对应总则中的 userId).和 添加的总则作用域(generalKinds)(改总则对应的航空公

司二字码/或者通用总则)去数据库查询 是否已存在该二字码。即异步校验其二字码唯一性。如果校验通过，则调用 service 里的 addGeneral 方法进行添加。同时数据要添加到缓存。

删除：删除总则不是真正的删除，是把一条总则的数据截止时间添加一个当前系统时间。数据依然存在数据库中，只是没做查询处理。当前登录用户只能删除与之 id 对应的总则。DataSecurityUtil.*validate*(session, *SKEY*, id)，运用的该统一验证校验 操作数据是否允许操作

修改：修改的本意是综合了增加和删除，增加成功，给原始被修改的总则的数据截止时间添加一个当前系统时间，修改调用的也是增加的方法 addGeneral。

当前登录用户只能修改与之 id 对应的总则。同删除一样调用的是该 DataSecurityUtil.*validate*(session, *SKEY*, id)方法去校验改数据是否可以被当前登录用户操作。

查询：从第一页开始显示，查询出所有的 当前登录用户的总则数据，处理了排序，数字、字母、汉字最后。

复制：复制模块的功能是，用户觉得其中一条总则数据适用于另外一个航空公司，就无须再去添加一条差不多的数据，只需复制该数据，修改其二字码就好。调用增加的 addGeneral 方法。复制同删除修改一样需 validate 校验权限。

前端页面校验：

主要的是同盟航司的校验、结算价计算方法的校验。

1、同盟航司校验：增加初始化页面时，只默认显示一组，一组一横排 4 个 input。每一个 input 在 js 里面都有校验需填写正确的二字码信息。

1.1 可动态添加更多组，也能删除更多组，同理每个 input 都进行了校验。由于这些 input 是动态添加的，引进的 aralejs 里的 validator.addRule()不能满足其需求，就需要在添加一组后 生效该校验。最后想到用，aralejs 里的 validator.addItem 方法，每添加一个 input objVali['alliance'+(th.createSequece()-1)]='required/allianceFormat/airlineExistAlli';就将这些规则放进一个数组 objVali[]里，添加一组后对数组里面的规则遍历，有需要必填的就返回 true.然后添加 rules[1],rules[2]两个校验规则。实现动态校验。

1.2 ，修改页面时，先计算该条总则有多少个同盟航司，由于增加存储时，是将所有的同盟航司存在一个字段中，组与组之间是用的”/”分开，组内每个之间用的是”-”分开。修改页面回显的时候，返回的是一个数组和一个字符串到页面，数组是直接放进 jsp 页面写死的第一组里面。字符串是存储同盟航司的字段，由 js 接收该字段进行分割，计算应该生成多少组 input 来进行存放这些二字码。

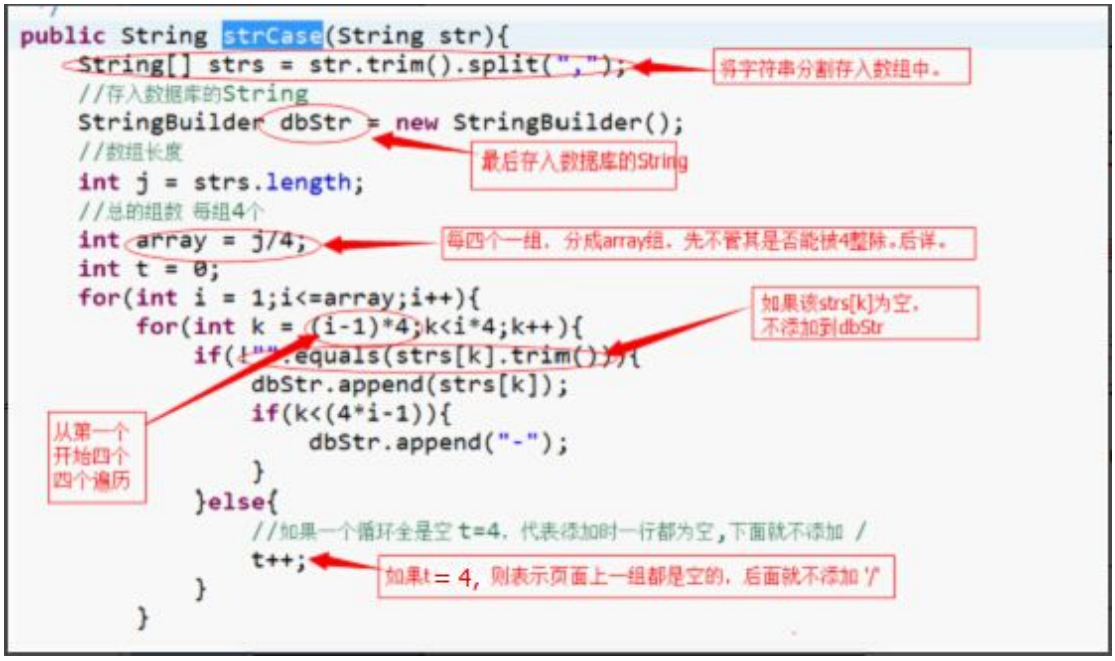
流程图如下：

首先是存储同盟航司的方法：GeneralService 类里面的 strCase 方法

传入的参数是 String str 存储的同盟航司。字符串在存储多个 input 类容时，默认是用“，”进行分割。先将其存入一个数组，遍历数组每一个，每个之间用“-”,每组之间用“/”append 到 dbStr 这个字符串。最后的 dbStr 格式应该是 类似这样的“CA-CZ-CM-AE/CN-AH/AR/AI-8C-3U/DZ-AV”字符串。

下面是该方法详解。

如图：



2、结算价计算方法的校验

结算方法一的校验：

主要实现的功能：

1、区间 1 到区间 3 是从小到大的数字填写校验.

校验规则命名为：rangeVal

区间上限校验：beginVal

2、点相应的按钮才可在空格中输入，实现的代码如图：

```
/**勾选才可填写**/
writeCase : function(father,child){
    $(father).change(function(){
       if($(father).attr('checked')==='checked'||
               $(father).attr('checked')==true){
           $(child).removeAttr('disabled');
       }else{
           $(child).attr('disabled','disabled');
       }

    });
    if($(child).val()==null||$(child).val()!=''){
        $(child).removeAttr('disabled');
    }else{
        $(child).attr('disabled','disabled');
    }
    if($(father).attr('checked')==='checked'||
            $(father).attr('checked')==true){
        $(child).removeAttr('disabled');
    }
},
```

两个参数，一个是按钮，一个是input框

结算方法二的校验：

这里算是一个页面上的难点校验，在 add.js 这个 js 文件中，和其他几个校验一样，没有用 validator 框架的方法，单独写的方法来实现该校验，主要实现的功能是如图

下一级会受到上一级的牵动，点击上一级，下级按钮自动选中，下级按钮关闭完，上级

按钮自动关闭。此校验规则实现的方法是 add.js 中的 caseEvent:function(parent,child){}方法，其中该方法中包括两个方法：

向下驱动方法：caseEvent:function(parent,child){}

与向上驱动方法： $(child).each(function(){}。

之后遇到一个问题，一个 jsp 页面出现了两个 validator 一个是引进的 Validator = require('validator');

一个是上一个为了实现动态校验 new 出来的

```
Var validator = new Validator(
    {element:'#generalAdd'}
);
```

而问题所在就是一个 jsp 中 只要这其中的一个 validator 校验成功就会给 页面最终返回 true ，即使另一个 validator 的校验规则没通过也能提交页面。

为了解决这个问题，最后把提交按钮改成了用 js 提交，在提交前，先查询每一个 input 的校验是否通过，如图：

```
/**提交**/
$('#buttonSubmit').bind('click',function(){
    var explainSize = 0;
    Widget.autoRenderAll();
    $('.ui-form-explain').each(function(){
        var inHtml = $(this).html();
        inHtml = inHtml.trim();
        if(inHtml !== '' && inHtml != undefined){
            $(this).focus();
            explainSize = explainSize + 1;
        }
    });
    if(explainSize == 0){
        if($('#generalKinds').val().trim()===''){
            $('#generalKinds').focus();
            $('#generalKinds').trigger("blur");
        }else{
            $(this).attr('disabled','disabled');
            $('#generalAdd').submit();
        }
    }
});
```

用来记录页面中input校验未通过个数

类选择器遍历input下的显示未通过字样的个数

如果有提示信息，说明未通过，焦点到该位置，并给explainSize+1

允许进入提交方法。

其中，`explainSize`用来记录个数，当等于零的时候，才允许提交。

3.2.7 模块的异常情况

本模块的异常主要出现在，参数填写错误带来的异常，这一块异常处理是取消当前用户的操作，并提示他。

3.2.8 模块尚未解决的问题

暂 无

3.3 国际政策批量导入详细设计

3.3.1 模块概述

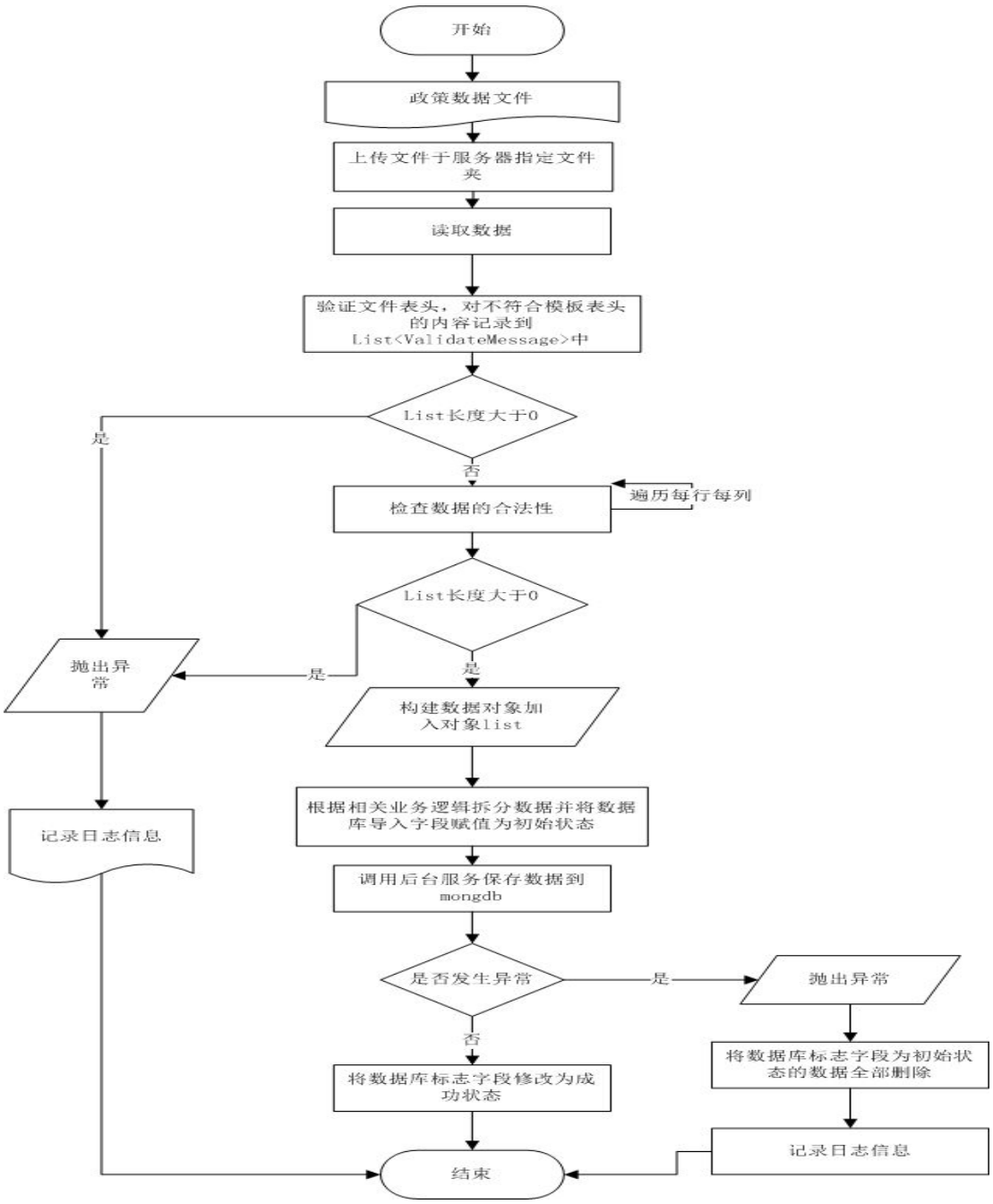
国际政策批量导入，主要是指用于根据系统指定的模板将大量的数据放入 excel 中，一次性导入到系统中，减少页面输入环节，提高政策添加效率。此模块主要涉及 excel 文件上传、数据格式的校验、入库等操作，另外系统在导入过程中实时将导入过程通过日志形式打印在前台页面，以供用户明确导入进程。

3.3.2 模块的需求描述

将一定格式的 excel 文档，在页面提交后，后台经过格式校验、数据解析、存入数据库，在此过程中前台实时打印处理日志。另外导入的数据必须和页面添加的政策数据保持一致。

3.3.3 模块的设计思路

根据需求，我们得出如下设计思路图：



根据上述图所示，我们的设计思路是：首先我们将用户上传的 excel 文件保存到服务器，接着解析服务器上的 excel 文档，在解析文档的过程中我们加入表头验证（保证用户 excel 必须和系统提供的一致，提高数据的完整性）、每一项的格式校验（保证数据的正确性），表在验证的过程中，如果出现异常或与模板的不一致时，将错误信息存入 ErrorQueue 的 MessageList 之中，当每一次大的验证完成之后，自动检测 ErrorQueue 的 MessageList 是否为空，如果时空，则继续，不然就抛出异常，打印错误日志给用户，如果校验没有错误，根据相关逻辑将数据在存入数据库之前进行包装，整合。最后将整合完成数据存入 mongodb 数据库，在此检测存储是否异常，如果有异常，回滚保存数据并打印错误日志给用户，如果

没有异常，修改标志字段，提示用户导入成功。

3.3.4 模块的数据库设计

这个模块不涉及到数据库设计，数据库设计由政策维护模块设计。

3.3.5 模块的限制条件

本模块对用户导入的 excel 有以下限制：

1 导入的模板必须在符合系统规定的格式，包括表头的名字也必须一致，也不能交换顺序。

2 导入的模板必须是以.xls 或.xlsx 的 excel 文件。

3 导入的数据需满足一下格式：

名称	类别	规格
基础信息		
出票航空公司	必填	两字码，多个用斜线分隔，小写自动转大写。
投放分销商	必填	自由文本，多个用斜线分隔
批发商		自由文本
旅客信息		
旅客身份		
旅客人数下限	必填	数字（3），默认 1
政策适用类型	必填	填写：散客，团队，多个用斜线分隔。
地理位置信息		
行程类型	下拉框	选项：OW、RT、OW/RT，默认 OW/RT
去程起点	必填	机场/城市三字码或中文名，多个用斜线分隔。
去程起点除外点		机场/城市三字码或中文名，多个用斜线分隔。
去程终点（折返点）	必填	机场/城市三字码或中文名，多个用斜线分隔。
去程终点（折返点）除外点		机场/城市三字码或中文名，多个用斜线分隔。
回程终点		机场/城市三字码或中文名，多个用斜线分隔。
回程终点除外点		机场/城市三字码或中文名，多个用斜线分隔。
日期要求		
第一国际段旅行日期	必填	yyyy-mm-dd，范围用~，多个斜线分隔。

最后一国际段旅行日期		yyyy-mm-dd，范围用~，多个斜线分隔。
出票日期范围	必填	yyyy-mm-dd，范围用~，
更多要求		
全程同一承运人		默认为空，即不选择
不允许代码共享		默认为空，即不选择
仅允许和代码共享		航空公司两字码，多个用斜线分隔。
必须直飞		默认为空，即不选择
不能经过		机场/城市三字码，多个用斜线分隔。
必须经过		机场/城市三字码，多个用斜线分隔。
排除航班号		两字码+航班号。 可以是单个航班或范围，*号表示一位任意数字，需要填写航空公司两字码，多个用斜线分隔。
仅限航班号		
政策舱位+返点		
舱位	必填	填写舱位字母，可以多个，斜线分隔
返点	必填	正数（一位小数），默认 0
舱位		填写舱位字母，可以多个，斜线分隔
返点		正数（一位小数），默认 0
舱位		填写舱位字母，可以多个，斜线分隔
返点		正数（一位小数），默认 0
手续费		正或负整数，默认 0
代理费		数字（2），最多保留一位小数
销售配置		自由文本
销售备注		字符串（250）

3.3.6 模块的代码设计

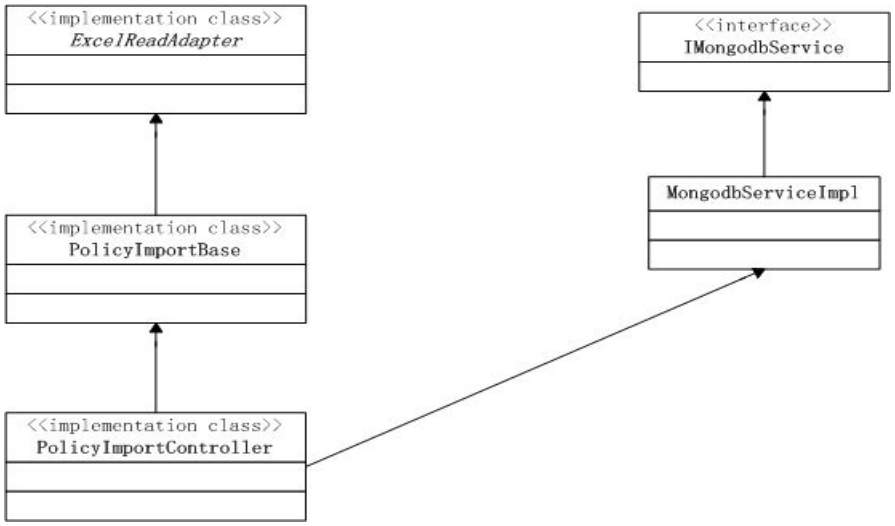
国际政策导入在设计时主要分成了 3 个模块：政策数据解析、校验、数据的校正、导入 mongodb。

1 在政策数据的解析和校验模块，定义一个 excel 通用的抽象类 ExcelReadAdapter<T>类

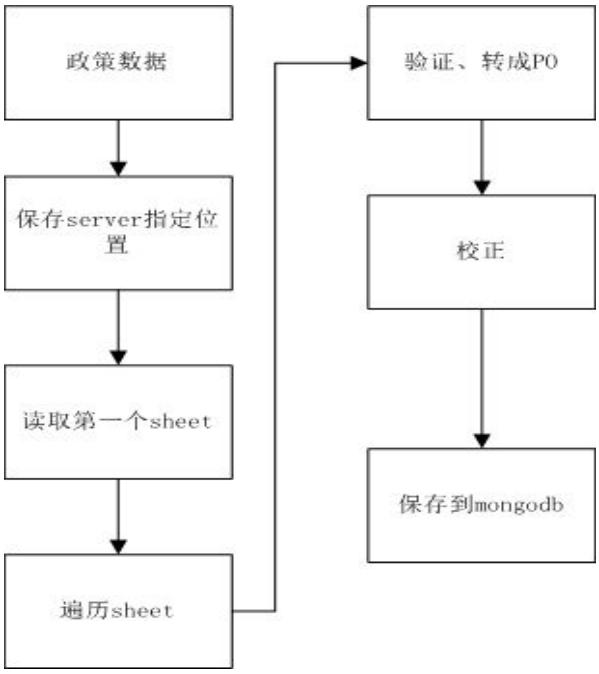
解析数据校验数据，并且将解析完成的数据加入政策对象 List 中，在具体的子类实现中定义 PolicyImportBase 来继承 ExcelReadAdapter<T>，具体的实现政策的验证、数据的校正。

2 在完成校验和数据的校正之后，调用 MongodbService 类完成对数据的批量导入。

下图为本模块的相关类的依赖关系。



导入的整个处理流程是，用户提交 excel 表格，服务器将用户提供的 excel 重新命名后保存到指定位置，然后读取解析 excel 的第一个 sheet，接着遍历 sheet 的每一行，每一列进行一下操作：获取数据->验证格式->装成对象属性，完成此操作之后校正数据，最后存入 mongodb。具体的如下图所示：



3.3.6.1 保存 excel 文件到指定的文件夹和读取 sheet

调用系统中工具类 FilesUtil 的 uploadExcel 方法完成 excel 的保存，并返回保存的路径，以供后续读取。如果保存有异常，直接中断响应，返回日志信息给用户。

调用 ExcelUtil 的 getSheetByExecl 的方法来读取 sheet。

3.3.6.2 遍历 sheet 和解析、校验数据

此模块的操作类是 PolicyImportBase，他继承了 ExcelReadAdapter，在 paserToPO 的方法中具体的实现的验证。在 analyticalData 方法中完成了数据的校正和初始化存入数据库的标志字段。

3.3.6.3 保存数据到 mongodb

这个模块主要是调用 MongodbServiceImpl 的 saveListObject 方法批量插入数据，完成之后更新标志字段，如果出现异常删除刚导入的数据。

3.3.7 模块的异常情况

导入模块对于系统级别的异常，在 Controller 层进程 try-catch，打印日志信息，给用户返回”系统异常“，在政策校验和校正中的异常会计入 ErrorQueue 的 messageList 中，返回给用户详细的提示。

3.3.8 模块尚未解决的问题

暂无

3.4 国际政策导出详细设计

3.4.1 模块概述

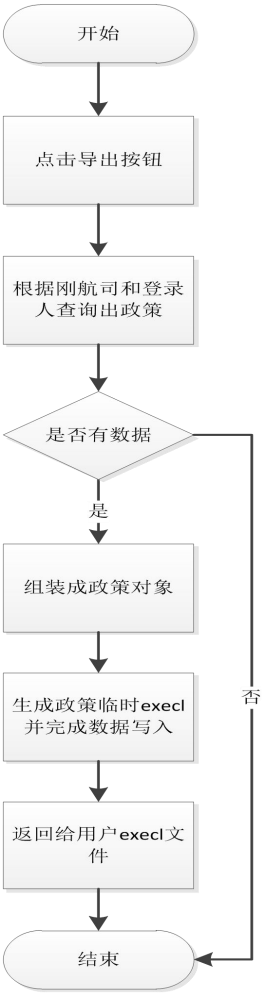
国际政策导出主要功能是根据航司以 execl 形式导出登录人的添加的政策。

3.4.2 模块的需求描述

政策维护人员进入政策导出页面，输入航空公司两字码，点击“开始导出”按钮。导出数据仅指当前政策库中所有有效的政策，已过有效期和未来有效的政策不能导出。待数据导出后，由浏览器弹出提示并保存 excel 文件。

3.4.3 模块的设计思路

根据需求，我们得出如下设计思路图：



根据上述图所示，用户点击导出按钮，系统根据条件查询出数据，组成成政策对象，写入到临时 execl 中，最后将 execl 文件传给用户。

3.4.4 模块的数据库设计

这个模块不涉及到数据库设计，数据库设计由政策维护模块设计。

3.4.5 模块的限制条件

无

3.4.6 模块的代码设计

本模块代码涉及以下主要类：

(1) PolicyImportController 此类中的 `public String export(String`

airLine, Model model,HttpServletResponse response, HttpSession session)主要接受用户导出政策的请求，并查询政策数据，调用政策对象组装方法组装成对象，在调用 execl 写入方法写成 execl 提供并传输给用户。

(2) ExcelUtil 类中的 public static void policyToXlsExcel(String modlePath, String outName,List<PolicyIntlPo> policyList) throws SysException 生成一个 execl 临时文件并调用对象组装方法 setPolicy(HSSFRow row, PolicyIntlPo policyIntlPo,HSSFCell cell, HSSFCellStyle style)来组装对象。

（1）FilesUtil 类中的 exprotExcel(String filePath,String fileName, HttpServletResponse response)throws SysException 导出文件给用户。

3.4.7 模块的异常情况

捕获未知异常，并将信息返回页面。

3.4.8 模块尚未解决的问题

暂无

3.5 国际政策匹配模块详细设计

3.5.1 模块概述

该功能用于根据 pricing 结果或 shopping 结果，进行政策的匹配，并计算出结算价，最终将 pricing 结果+结算价（政策内容），或 shopping 结果+结算价（政策内容）一并返回给客户。这个模块仅是根据输入内容匹配政策以及计算结算价。

用户调用 shopping+政策接口，或 pricing+政策接口，或 PNR+政策接口，之后先调用 shopping 接口，或 pricing 接口，或 PNR 计算接口，拿到计算结果后，再拼装成政策请求，进行政策匹配和结算价的计算，最终将政策结果和计算结果一起返回给用户。

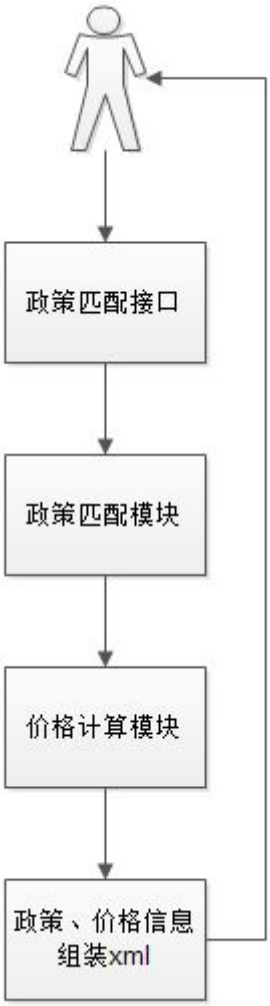
3.5.2 模块的需求描述

该功能用于根据 pricing 结果或 shopping 结果，进行政策的匹配，并计算出结算价，最终将 pricing 结果+结算价（政策内容），或 shopping 结果+结算价（政策内容）一并返回给客户。这个模块仅是根据输入内容匹配政策以及计算结算价。

用户调用 shopping+政策接口，或 pricing+政策接口，或 PNR+政策接口，之后先调用 shopping 接口，或 pricing 接口，或 PNR 计算接口，拿到计算结果后，再拼装成政策请求，进行政策匹配和结算价的计算，最终将政策结果和计算结果一起返回给用户。

3.5.3 模块的设计思路

根据接口返回的 xml，请求国际政策匹配接口，匹配出政策相关信息，并根据政策信息计算出相关价格，最后组装成 xml 返回给客户端，设计思路图如下：



从上图可知，政策匹配模块又分为政策匹配和价格计算两个子模块，下面会详细介绍。另外为了方便描述，下面以 shopping 政策匹配为例，描述政策匹配模块的流程。

3.5.4 模块的数据库设计

这个模块不涉及到数据库。

3.5.5 模块的限制条件

针对政策匹配接口传入的参数和返回的结果，下面进行简要说明。

3.5.5.1 接口传入参数说明

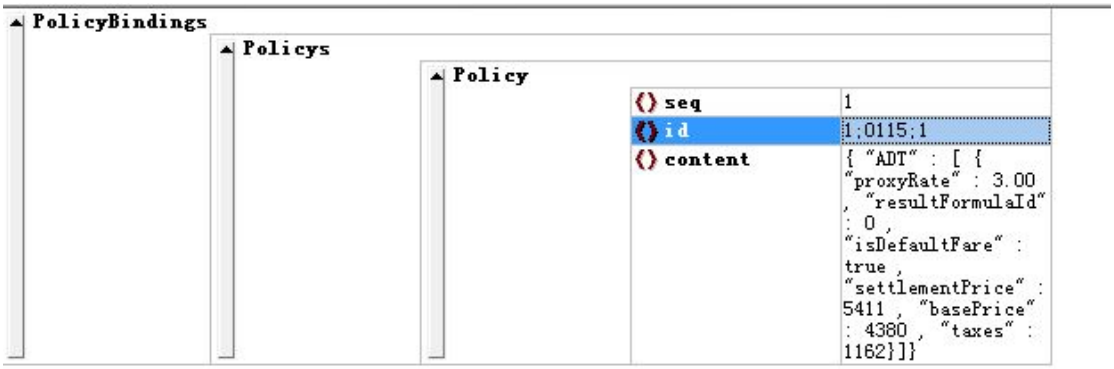
输入参数	参数说明	是否	数	举例	备注

		必填	据类型		
基础信息 （一个 shopping 结果仅有一个基础信息，一个 pricing 结果也仅有一个基础信息。）					
分销商信息	分销商集合名称，和政策数据中一致	是			直接和政策中相应字段进行匹配。
出票时间	通过出票时间匹配此出票时间对应的有效政策。如果是历史政策匹配，需传入历史出票时间。	是		2014-3-11 19:00	Shopping 和 Pricing 只传一个出票时间。当出票日期早于当前日期时，认为是历史匹配。
拼散包团	标识查询政策是查散客还是团队，默认散客。	是		散客/团队	
旅客信息（多人） （一个 shopping 结果仅有一组旅客信息，一个 pricing 结果也仅有一组旅客信息。）					
旅客类型代码	旅客类型三字码	是		ADT	同 shopping 或 pricing 接口里的旅客类型代码
旅客人数	实际行程中每个旅客类型的人数，默认为 1。	是		2	同 shopping 或 pricing 接口里的旅客数量
航段信息（每一个航段有如下参数；至少一个航段，可以有多个） （一个 pricing 结果对应一组航段信息，一个 shopping 请求有多个结果，每个结果都对应一组航段信息）					
序号	航段标识	是		1、2、3、4...	
出发机场	机场三字码	是		PEK	
到达机场	机场三字码	是		LHR	
出发日期	起飞的日期	是		2014-4-18	
出发时间	起飞的时刻	是		11:20	
到达日期	降落的日期	是		2014-4-18	
到达时间	降落的时刻	是		19:15	
市场方	市场方航空公司两字代码	是		BA	
承运方	承运方航空公司两字代码，不填则认为与市场方一致。	否		BA	

航班号	航班号	是		CA991	航空公司+航班号
舱位	舱位代码	是		B	注：Shopping 应该是从 FareInfos 里的 FlightResBookDesigCode 字段获取各航段的舱位，不能是 AV 的舱位
里程数	里程数，单位英里。	否		5048	Shopping 有 TPM 里程数，但是没有 MPM 里程数。Pricing 接口都没有。
票价信息（每个结果对应一个票价信息） （一个 pricing 结果对应一个票价信息，一个 shopping 请求有多个结果，每个结果都对应一个票价信息）					
出票航空公司	TicketingAirline，两字码。	是		CA	
票面价	票面价格，不含税，BaseFare。	是		2600	
税款	税费总金额。	是		1249	
Q 值	附加费用。	否		40	
代理费	代理费率	否		3	
计算横式	文本	是		Bjs ca hkg 235.23yow nuc235.23end roe6.244500	
FC 信息（至少有一个，可以有多个） （一个 pricing 结果对应一组 FC 信息，一个 shopping 请求有多个结果，每个结果都对应一组 FC 信息）					
FC 起点	城市三字码	是		BJS	
FC 终点	城市三字码	是		LON	
Fare Basis	票价基础	是		YOW	
FC 票价航空公司	FareOwner，对应字段名叫 FilingAirline。	是		BA	
FC 对应航段序号	标识每个 FC 对应哪几个航段信息。	是		123	

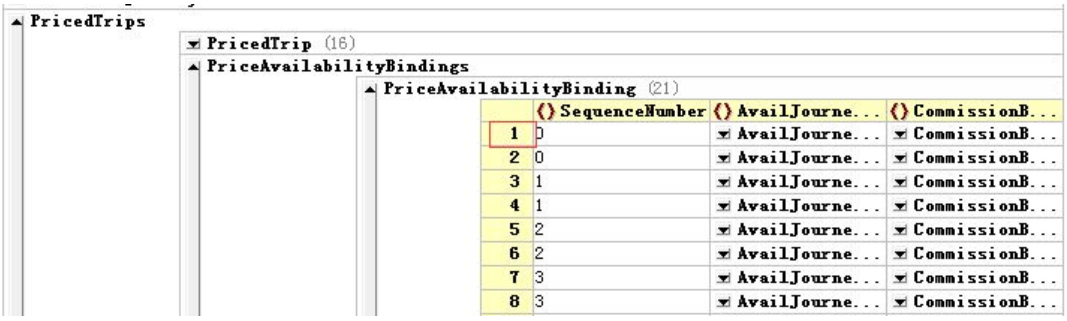
3.5.5.2 接口返回参数说明

返回的 xml 结构如下：

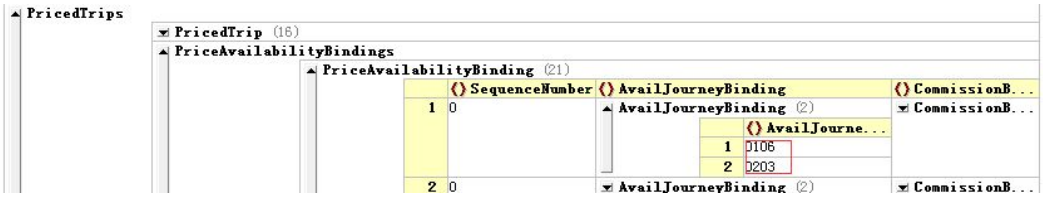


Policy 节点中包含 seq、id、content 节点，这 3 个节点都必须有值。

Seq 节点：其值和下图红框中的值一一对应（下图为 shopping 结果 xml 相关字段，pricing、PNR 导入的类似）。



Id 节点：其值的组成为下图红框中的值和 seq 号组成，如下图的例子，则 id 节点的值为：0;0106;0203;1，其中 0 为 sequenceNumber，然后是“;”，0106 为下图框中的第一个值，然后是“;”再是下图框中的第二个值，然后是“;”最后是上节描述的 seq 节点的值。这个节点主要用户调试问题，用户可以不用关注。



Content 节点：content 节点里的字段顺序是不一定的，可能出现的字段和值情况见下表说明。

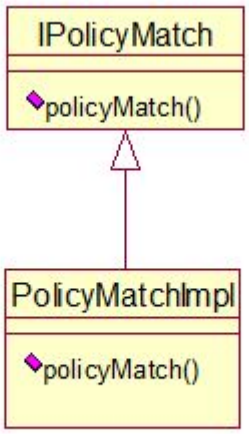
字段名	是否必填	字段说明
ADT	Y	用户类型，如果查询 shopping 时指定了多个用户类型，那么这里会出现多个用户类型

以下字段为某个用户类型匹配到的某个政策内容，如果没有匹配到政策，则没有以下政策内容。		
政策内容		
_id	Y	政策 id
tktAirline	Y	出票航司
distributor	Y	投放分销商
wholesaler		批发商
passagerType	Y	旅客类型
passagerCount	Y	旅客人数
groupType	Y	团队类型
travelType	Y	行程类型
outOrg	Y	去程起点
outOrgEx		去程起点除外
outDes	Y	去程终点
outDesEx		去程终点除外
returnDes		回程终点
returnDesEx		回程终点除外
firstTravelDate		第一国际段旅行日期
lastTravelDate		最后一国际段旅行日期
startTktDate	Y	出票日期开始
endTktDate	Y	出票日期结束
isCarrierOne		是否同一承运人
isNoAlloewShareCode		是否不允许代码共享
allowShareCode		允许和谁代码共享
isDirectFlight		是否必须直飞
noAllowPass		不允许经过
mustPass		必须经过
allowFilghtNo		仅限航班号
excludeFilghtNo		排除航班号

cabinOne	Y	舱位一
rebateOne	Y	返点一
cabinTwo		仓位二
rebateTwo		返点二
cabinThree		仓位三
rebateThree		返点三
mixCabin		混舱返点选择
poundage		手续费
proxyFree		代理费
sellConfig		销售配置
remark		备注
status	Y	状态
workDate	Y	生效时间
endDate		生效截止时间
importKey		本次导入编号
userId	Y	用户 ID
以下是结算价格相关内容，不管是否匹配上政策，这些都是必填内容		
basePrice	Y	基础运价，来源于 shopping 结果
taxes	Y	其他税款，来源于 shopping 结果
proxyRate	Y	代理费
settlementPrice	Y	结算价
isDefaultFare	Y	是否是默认政策
resultFormulaId	Y	使用的公式 Id

3.5.6 模块的代码设计

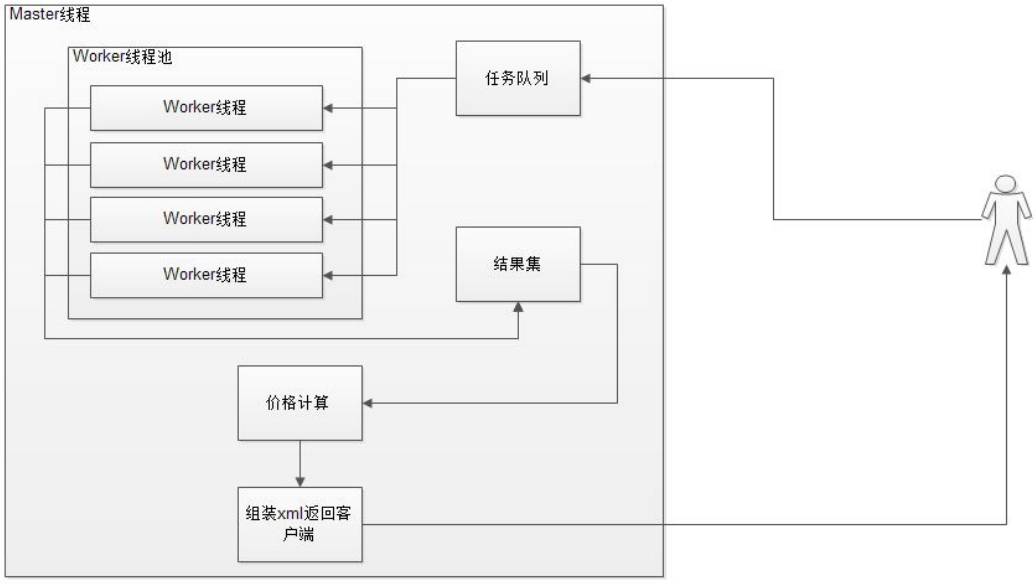
在调用政策匹配接口 IPolicyMatch 时，其实现类为：PolicyMatchImpl。类图情况如下：



在 PolicyMatchImpl 类的 policyMatch 方法定义如下：

public String policyMatch(List<FareRequest> fareRequest) throws AppException

在政策匹配模块，包含政策匹配子模块和价格计算子模块，并且政策匹配接口方法的参数为 list 集合，且接口方法返回的政策信息条数和请求参数集合的条数要一致。为了提高匹配效率，在这里使用了 Master-Worker 模式，如下图所示：



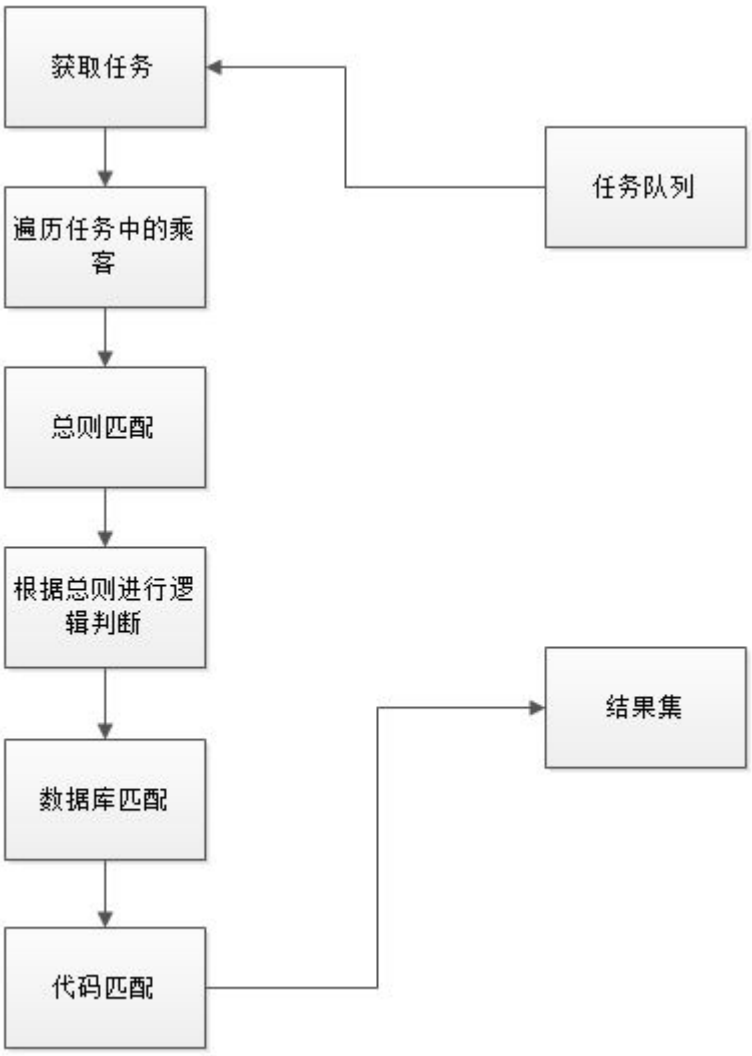
在上图中。Master 线程即为用户请求政策匹配接口线程（主线程），在主线程的接口方法 policyMatch 中，我们将请求集合存储到任务队列中，然后调用 worker 线程池开启 N 个线程对任务队列进行政策匹配，并把匹配到的政策信息存储到结果集中；主线程则从结果集中拿出匹配到的政策进行价格计算，当任务队列被政策匹配完且结果集中的政策都被价格计算完成，则将所有的结果进行组织成 xml，并返回给客户端。

在代码中，任务队列和结果集被定义到 TheadHandleData 对象中。Worker 线程池定义在 ThreadPoolService 对象中，具体的 work 线程为 FareQueryThread 对象，master 线程为 PolicyMatchImpl 类。

下面分别从 worker 线程（政策匹配）和 master 线程（价格计算）两方面进行详细介绍。

3.5.6.1 政策匹配子模块

政策匹配子模块从 FareQueryThread 线程的 run 方法开始，首先从任务队列里取出要匹配的运价信息（取出一条记录就删除一条记录），然后进行相关数据的设置，最后调用 queryPolicy 方法处理。政策匹配子模块的处理逻辑如下图所示：



queryPolicy 方法主要处理的是遍历任务中的乘客，并调用总则匹配和根据总结进行逻辑判断，并根据逻辑判断结果调用数据库匹配。

3.5.6.1.1 总则匹配

总则匹配是在政策匹配之前的一个步骤。它的作用就是通过传入的 Office（一代信息）和出票航司找出每个代理人在平台事先配置好的总则信息，通过获得的总则信息去获得判断行程的 ODD 点、行程类型的方法，以及同盟航司、计算结算价方式等信息；

系统中的总则信息已经事先按照用户 id+航司作为 key，总则信息作为 value 的键值对存入到了 redis 缓存中。在政策匹配时，总则的匹配过程是：

首先，将接口中的 office 号（一代信息）在本平台信息的用户信息中查找到 office 对应的用户信息（这一步也是在 redis 缓存中实现的，office 号作为 key，用户信息作为 value）。

然后，用用户 id 和出票航司作为 key 去 redis 缓存中查找总则信息；

如果没有找到，则会去缓存中查找该用户的通用总则，如果仍然没有找到，则抛出应用异常，是其走默认政策。

最后将匹配到的总则返回给调用者。

总则匹配的类 GeneralVoMatchServiceImpl，其方法是 public GeneralVo match(GeneralMatchParams generalMatchParams)。

3.5.6.1.2 根据总则进行逻辑判断

得到总则后，接下来就是根据总则的信息进行一些处理。

说明：上一步的总则匹配和该步骤中的逻辑处理，入口是：IPolicyMatchLogicService 接口的 public PolicyLogicResult doLogic(PolicyLogicParams request) throws AppException 方法；

1）判断是否符合无奖励条件，IPolicyNoAwardService 接口的 NoAwardResults isNoAward(NoAwardParams noAwardParams)方法

如果满足无奖励条件，则结束逻辑判断，否则进行下面的处理；

2）获得航线中的所有航班号（市场方+航班号）；

3）ODD 点设置，根据总则的中的设置判断出 ODD 点，IOddService 接口主要完成该逻辑判断；

4）判断 ODD 点是否合理，在 PolicyMatchLogicServiceImpl 类中的 private void judgePoint(PolicyLogicParams request,PolicyLogicResult policyLogicResult) throws

AppException 方法实现；主要判断去程起点、去程终点（则返点）和回程终点三点是否出现错位的现象，如果错位则抛出应用异常，使其走默认政策；

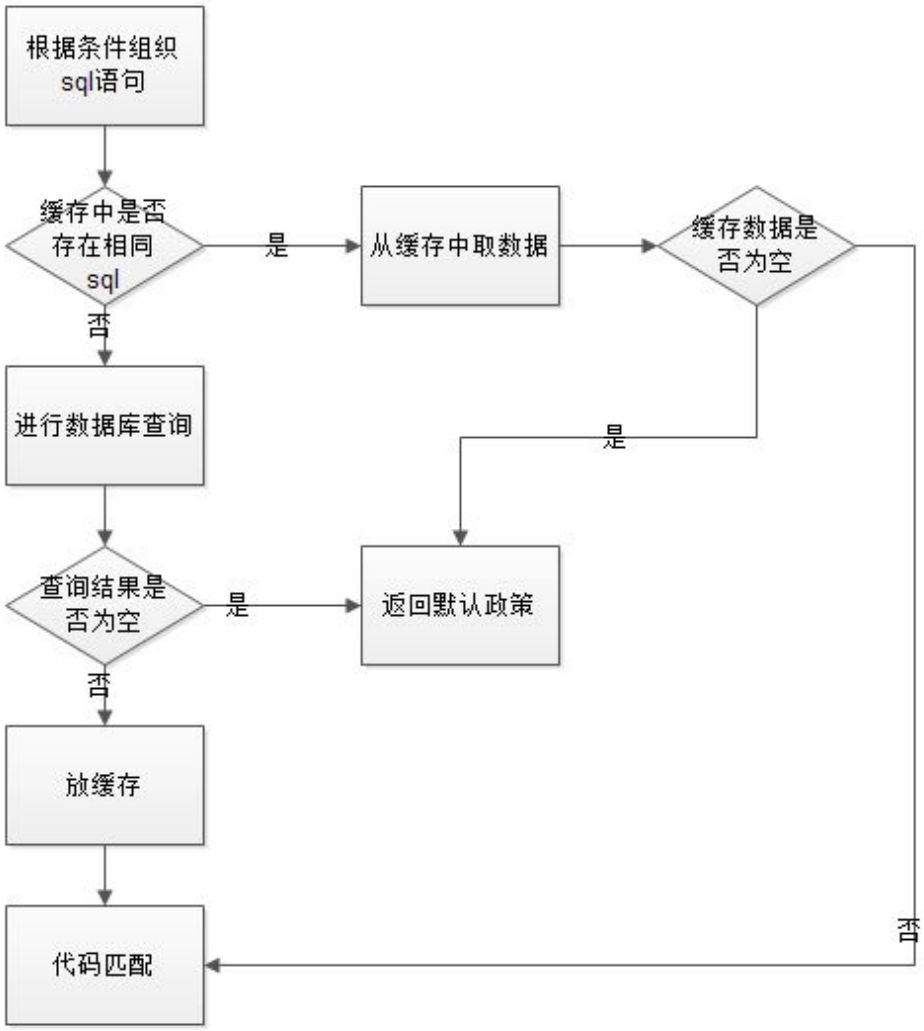
5）判断主航段和行程类型，private void setMFLAndTripType(PolicyLogicParams request,PolicyLogicResult policyLogicResult) throws AppException 方法，如果没有找到主航道，则抛出应用异常，使其走默认政策；

6）判断全程是否同一承运人和是否直飞，该部分在 IPolicyOtherLogicService 接口中实现。

7）初始化国际航段，将所有航段中的国际航段设置到返回结果中，供后面的匹配使用。

3.5.6.1.3 数据库匹配

数据库匹配，主要是根据条件，组织查询条件，在去数据库中进行查询的过程。是在 MatchingPolicyNoSqlService 类的 matchingFare 方法中进行的。该方法的逻辑如下图所示：



Sql 包含的查询条件如下：

名称	匹配逻辑	备注
基础信息		
用户 ID		
出票航空公司	用接口输入的“出票航空公司”进行匹配	匹配字段
投放分销商	用接口输入的“投放分销商”进行匹配，政策中可能有多个（斜线分隔），匹配一个即可。	匹配字段
旅客信息		
旅客类型	直接根据接口输入内容匹配	匹配字段
旅客人数下限	要求接口输入的人数大于等于此项	匹配字段
政策适用类型	根据接口输入匹配，政策中的/表示或（如果接口没有传入，则默认散客）	匹配字段
地理位置信息		
行程类型	根据总则逻辑判断的结果。	匹配字段
去程起点	根据总则逻辑判断的结果进行匹配。需要获	匹配字段

	取地理位置基础信息进行判断，要求包含在政策数据的位置里。	
去程起点除外点	根据总则逻辑判断的结果进行匹配。需要获取地理位置基础信息进行判断，要求不能落在政策数据的位置里。	匹配字段
去程终点	同“去程起点”	匹配字段
去程终点除外点	同“去程起点除外点”	匹配字段
回程终点	同“去程起点”	匹配字段
回程终点除外点	同“去程起点除外点”	匹配字段
日期要求		
政策适用日期范围		匹配字段
出票日期范围	根据接口输入的出票日期进行校验。	匹配字段
其他要求		
全程同一承运人	要求接口输入的所有航段的承运人都是出票航空公司。	匹配字段
必须直飞	如果是 OW，只能有一个航段； 如果是 RT，只能有两个航段。	匹配字段
不能经过	所有航段的起点和终点都不能是这个字段的内容。	匹配字段
必须经过	所有航段的起点和终点中，至少有一个点是这个字段的内容。	匹配字段

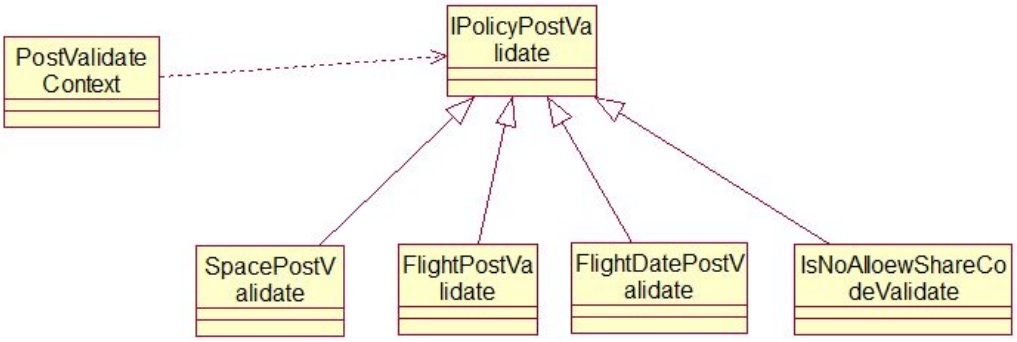
查询条件具体见 InitMatchingFareNoSql 类的 initMatchingFareSql 方法。

3.5.6.1.4 代码匹配

代码匹配主要是针对有些条件，无法方便的写入 sql 里进行查询的条件的过滤，主要包括仓位匹配、是否共享匹配、航班号匹配、航班日期匹配几个条件的匹配。

代码匹配的代码在 MatchingPolicyNoSqlService 类的 filterOtherRequest 方法里，主要是遍历 sql 查询的结果，再进行仓位匹配、是否共享匹配、航班号匹配、航班日期匹配。具体见代码。

在 filterOtherRequest 方法里实现代码匹配主要是调用了 PostValidateContext 类的 handlePostValidate 方法。PostValidateContext 类可以看成是策略模式的环境类，相关类图如下图所示：



在 PostValidateContext 类中的包含 List<IPolicyPostValidate> list 这个属性，该集合包含了上述几个匹配对象，在 handlePostValidate 方法中，将遍历该集合，调用该集合中的对象的 postValidate 方法。

1、仓位匹配：对应的类为 SpacePostValidate，业务逻辑为：

政策返点信息		
舱位	根据去程主航段、回程主航段的舱位来匹配。	匹配字段
舱位	根据去程主航段、回程主航段的舱位来匹配。	匹配字段
舱位	根据去程主航段、回程主航段的舱位来匹配。	匹配字段

即任务队列中的当前任务对应的去回程主航段的仓位是否被包含在政策中的这 3 个仓位集合中，如果是则匹配成功，否则该政策匹配失败。

2、是否共享匹配：对应的类为 IsNoAlloewShareCodeValidate，业务逻辑为：

不允许代码共享	要求市场方是出票航空公司的航段，其承运方也必须是出票航空公司。 市场方不是出票航司的航段不用校验。	匹配字段
仅允许和____代码共享	要求市场方是出票航空公司的航段，其承运方也必须是出票航空公司或此字段的航空公司。 市场方不是出票航司的航段不用校验。	匹配字段

如果当前政策中是否不允许代码共享的值为 true，表示不允许代码共享，即要求当前任务中市场方是出票航司的航段，其承运方也必须是出票航司，市场方不是出票航司的航段则不用校验。

如果当前政策中是否不允许代码共享的值为 false，表示允许代码共享，这需要和政策中仅允许和谁进行代码共享进行匹配。即要求当前任务中市场方是出票航司的航段，其承运方也必须是出票航司或此字段的航空公司，市场方不是出票航司的航段则不用校验。

3、航班号匹配：对应的类为 FlightPostValidate，业务逻辑为：

排除航班号	所有航段的航班号都不能是这个字段的内	匹配字段

	容。 政策中的/表示和	
仅限航班号	所有航段的航班号中至少有一个满足这个字段的内容。 政策中的/表示或。	匹配字段

政策中航班号的组成例如为：CA111-CA222/CA3***/CA345 这种形式。

其中仅限航班号指当前任务中所有航段的航班号中至少有一个满足这个字段内容，排查航班号则为当前任务中所有航班号都不能是这个字段的内容。

4、航班日期匹配：对应的类为 FlightDatePostValidate，业务逻辑为：

第一国际段旅行日期范围	第一个国际段的旅行日期必须在这个区间。中间可以有多个范围，用/分隔。	匹配字段
最后一国际段旅行日期范围	最后一个国际段的旅行日期必须在这个区间。中间可以有多个范围，用/分隔。	匹配字段

政策中的航班日期组成例如为：20140101-20140601/20140909 这种形式。

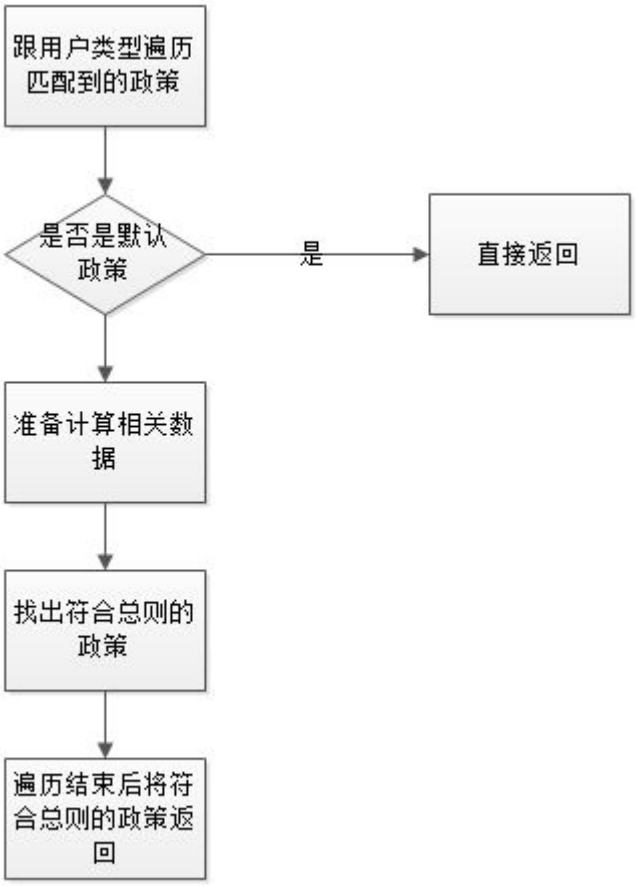
其中第一国际航段日期范围字段是指当前任务的第一个国际航段的旅行日期必须在这个字段的范围内，且最后一国际航段旅行日期范围是指当前任务的最后一个国际航段的旅行日期必须在这个字段的范围内。

当所有的代码匹配完成后，如果由于匹配到政策，则把这些政策放进结果集中。

3.5.6.2 价格计算子模块

主线程即政策匹配接口方法，调用 handleFareData 方法进行价格计算，该方法从结果集中取出匹配到的数据，根据用户类型进行遍历，调用 DataHandle 类的 dataHandle 方法进行价格计算。

dataHandle 方法的处理逻辑如下图所示：

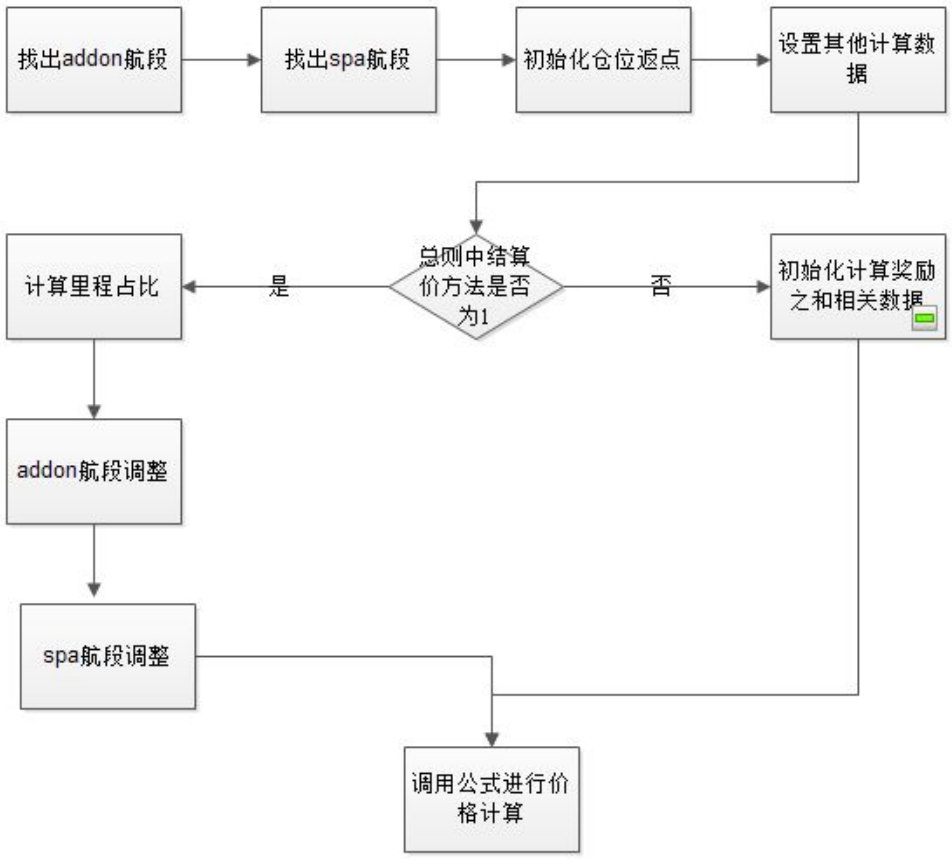


上图中如果是默认政策，则直接返回，因为默认政策已经将价格按照默认公式进行计算完毕，不需要再进行价格计算，具体见默认政策处理逻辑。

这个章节重点应该关注准备计算相关数据子模块和找出符合总则的政策子模块。

3.5.6.2.1 准备价格计算相关数据

价格计算从 DataHandle 类的 initExprission 方法开始，处理逻辑如下：



1、找出 addon 航段

Addon 段：航段的起点和终点都在中国。其中：

自营的 Addon 段: Addon 航段的市场方 AND 承运方 = 出票航（同盟）。

非自营的 Addon 段：Addon 航段的市场方≠出票航（同盟）或 承运方≠出票航（同盟）。

注：程序实现时，可以先找出所有的 addon 航段，其中市场方=承运方=出票航（同盟）的为自营的 addon 航段，剩余的都是非自营的 Addon 航段。

具体代码见 FlightLegParase 类的 paraseAddonFlightLeg 方法。

2、找出 spa 航段

SPA 段：航段的起点和终点都不在中国且不是主航段。其中:

自承运的 SPA 段: SPA 航段的承运方是出票航（同盟）。

与主航段同一票价计算组：和主航段在一个 FC 里。

与主航段不同票价计算组：和主航段不在一个 FC 里。

非自承运的 SPA 段: SPA 航段的承运方不是出票航（同盟）。

与主航段同一票价计算组：和主航段在一个 FC 里。

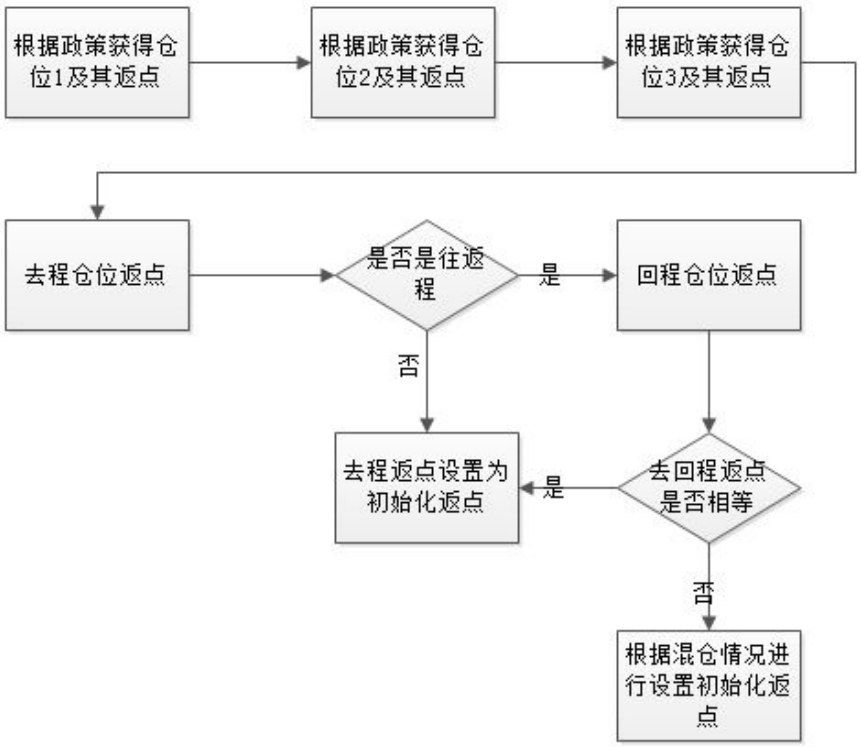
与主航段不同票价计算组：和主航段不在一个 FC 里。

注：无论是去程主航段，还是回程主航段，只要和其中一个在同一 FC 就行。

具体代码见 flightLegParase 类的 paraseSpaFlightLeg 方法。

3、初始化仓位返点

具体到代码为 InversePointHandle 类的 initInversePoint 方法，其逻辑如下图所示：



4、设置其他计算数据

主要是为计算准备相关数据，比如票价数据、代理费数据、税款数据、手续费数据，这步主要就是 get、set 的过程。

如果总则中的结算方法为 1，则需要进行如下调整：

非实际承运里程占比　区间 1__　返点调整_____；区间 2__　返点调整_____；

Addon　1 段　返点调整_____；　2 段 返点调整_____；

SPA　1 段　返点调整_____；　2 段 返点调整_____；

公式：结算价=票价*（1-代理费）*（1-返点）+ 税款 + 手续费；结算方法 1 主要是根据返点来计算结算价。

5、计算非实际承运里程占比

代码在 InversePointHandle 类的 calculateAddonNotVCPercent 方法。如果市场方不是出票方或联盟，那这一航段为非 VC 实际承运，如果市场方是出票方或联盟，但承运方不是出票方或其联盟，那这一航段也是非 VC 实际承运。里程占比主要是根据所有航段里的非 VC 实际承运航段里程数除以总里程数，看这个比例落在当前总则里的非实际承运里程占比的那一个区间，如果在区间 1，则返点加上区间 1 对应的返点。

6、addon 航段返点调整

Addon 航段数落在总则中关于 addon 航段调整，比如 addon 航段数为 1，曾将返回加上此区间对应的返点。

7、spa 航段返点调整

和 addon 航段返点调整类似。

8、初始化计算奖励之和相关数据

具体代码在 SumRewardHandle 类的 initSumReward 这个方法。

结算价方法 2 - 调整记入奖励的部分

计入奖励的航段是：

默认主航段计入奖励。

Add On 计入奖励

自营的 Addon 段计入奖励

非自营的 Addon 段计入奖励

SPA 计入奖励

自承运的 SPA 段计入奖励

与主航段同一票价计算组计入奖励

与主航段不同票价计算组计入奖励

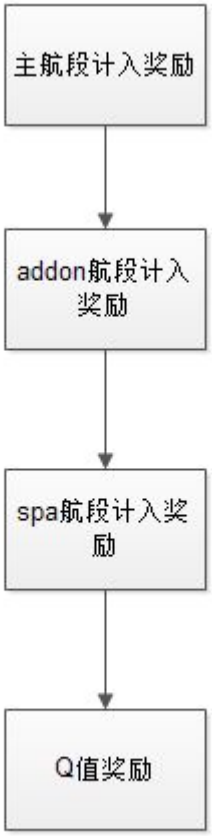
非自承运的 SPA 段计入奖励

与主航段同一票价计算组计入奖励

与主航段不同票价计算组计入奖励

Q 值计入奖励

公式：结算价=（记入奖励之和）*（1-代理费-返点）+ （票价-记入奖励之和）*（1-代理费） + 税款 + 手续费；结算方法 2 主要是根据计入奖励的方式进行结算。



主航段计入奖励、addon 航段计入奖励、spa 航段计入奖励，本身都比较简单，具体看相关代码，这里需要重点描述航段计入奖励的计算方法 paraseFareLinear，三种航段计入奖励都调用了该方法。

在讲解 paraseFareLinear 方法前，需要了解横式相关业务，具体见需求文档的解析横式相关章节。paraseFareLinear 方法实现的逻辑如下例子的逻辑一致：

举例：

<UnstructuredFareCalc>bjs 3u x/ckg 3u syd320.28e6maus 3u x/ckg 3u bjs320.28e6maus d ckgsyd640.56d ckgsyd640.56nuc1921.68end roe6.244500</UnstructuredFareCalc>

通过政策接口输入，已知：

FC1：对应航段 1/2，farebasis 是 E6MAUS，

FC2：对应航段 3/4，farebasis 是 E6MAUS

票价是 12000CNY，无 Q 值。

解析横式：

1 定位 farebasis 并得到每个 FC 的 NUC 价格

FC1：320.28;

FC2：320.28;

2 假设航段 2 需要计入奖励，航段 2 在 FC1 中。

假设各航段里程为：bjs-ckg:100, ckg-syd:900.

那么航段 2 的 NUC 价格为：320.28*900/(100+900) = 288.25

最终得到：

航段 2 价格 = 票价 * 航段 2NUC 价格/横式 NUC 总价

= 12000 * 288.25/1921.68 = 1800

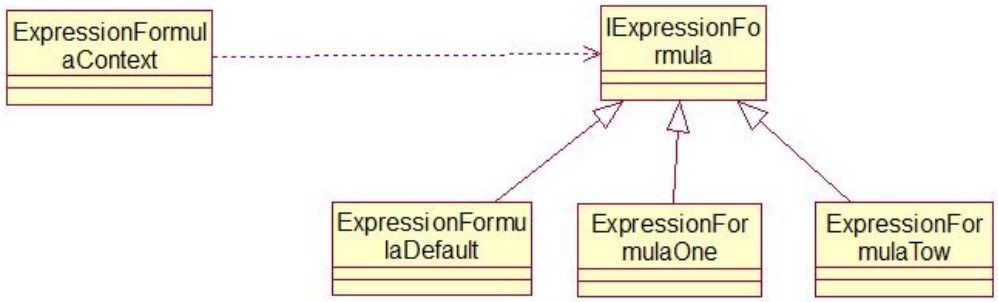
最终得到航段 2 的计入奖励为 1800，其他航段计入奖励的计算类似。

9、调用公式进行计算

这个功能是在 DataHandle 类的 finallyCalculate 方法中完成的。

该方法首先根据准备的数据和价格计算公式对结算价进行计算，如果结算价为负数，则调用默认政策的价格计算。最后将相关数据写入到返回值中。

这里价格计算掉用了 ExpressionFormulaContext 类的 expressionFormula 方法。具体类图如下：



ExpressionFormulaDefault 为默认政策价格计算类，公式为：默认政策 F*(1-C)+T。

ExpressionFormulaOne 为结算方式 1 对应的价格计算类，公式为：结算价=票价*（1-代理费）*（1-返点）+ 税款 + 手续费。

ExpressionFormulaTow 为结算方式 2 对应的价格计算类，公式为：结算价=（记入奖励之和）*（1-代理费-返点）+ （票价-记入奖励之和）*（1-代理费） + 税款 + 手续费。

计算过程比较简单，具体看相关代码。

3.5.6.2.2 找出符合总则的政策

当政策结算价计算完毕，需要根据该政策的结算价进行过滤，具体需求是根据总则中数据选取方式决定。

数据选取方式

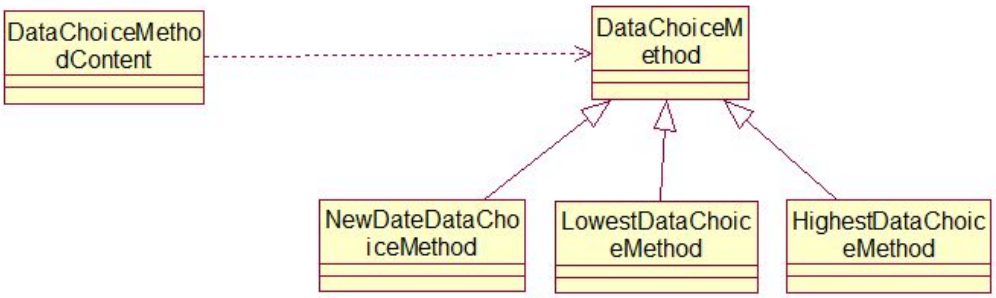
多条匹配时，最终仅选取 1 条。

仅返回数据日期最新的 （默认）

仅返回结算价最高的

仅返回结算价最低的

如果当前任务只匹配到一个政策，那么直接返回该政策及其结算价，如果匹配到了多条政策，则需要根据总则的数据选取方式，过滤出符合总则的那条政策及其结算价，并将其返回。这个过程主要在 DataChoiceMethodContent 类的 dataChoiceMethod 方法完成，具体类图如下：



这块逻辑比较简单，具体见相关代码。

3.5.6.2.3 默认政策价格计算

默认政策价格计算是指在政策匹配过程中或价格计算过程中出现异常或者政策匹配时未匹配到政策时，需要给用户返回默认政策价格结算价。代码主要在 DefaultPolicyHandle 类的 defaultFare 方法中。这个模块是政策匹配中的重要模块之一，但因为逻辑简单，这里不做详细介绍，详细情况见代码。

3.5.7 模块的异常情况

本模块在匹配子模块和价格计算子模块时，都分别对可能出现的异常进行了 catch，具

体代码见 FareQueryThread 的 queryPolicy 方法和 PolicyMatchImpl 的 handleFareData 方法。之所以要对异常进行 catch，是因为 shopping 结果可能有多个运价信息需要进行政策匹配，如果某一个运价信息的政策匹配过程出现异常，不能影响到其他运价信息的政策匹配，再加上返回给用户的必须为 xml 结构的字符串。如果在政策匹配过程中出现了异常，则执行默认政策价格计算。

3.5.8 模块尚未解决的问题

暂无

3.6 国际 Pricing 模块详细设计

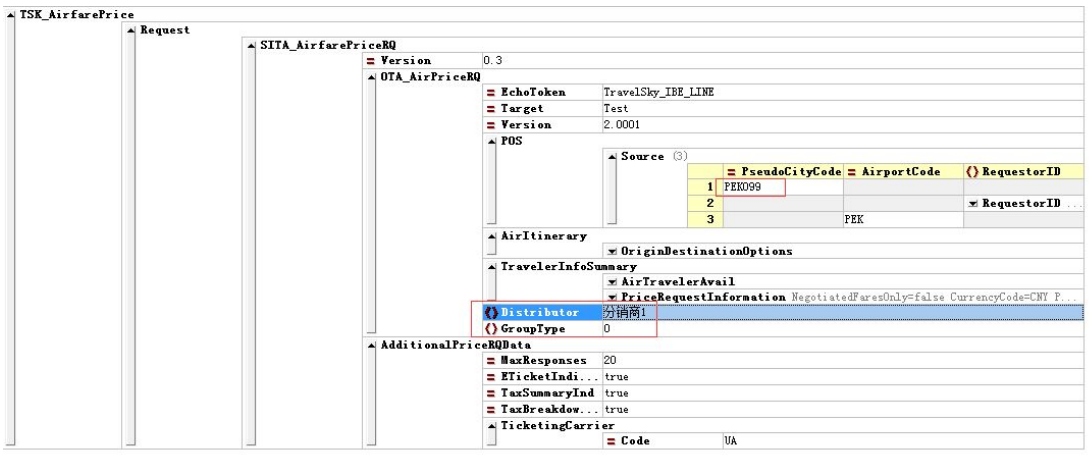
3.6.1 模块概述

国际 pricing 模块，是指用户调用我们对外发布的国际 pricing 接口时，我们根据用户请求的航段，调用 IBE+底层国际 pricing 接口，获得航段运价信息，并根据航段运价信息调用我们自己的国际政策匹配接口，获得运价信息对应的政策信息，并将运价信息和政策信息返回给用户。至于如何进行政策匹配将在政策匹配模块进行介绍。

3.6.2 模块的需求描述

国际 pricing 模块，是指用户调用我们对外发布的国际 pricing 接口时，我们根据用户请求的航段，调用 IBE+底层国际 pricing 接口，获得航段运价信息，并根据航段运价信息调用我们自己的国际政策匹配接口，获得运价信息对应的政策信息，并将运价信息和政策信息返回给用户。

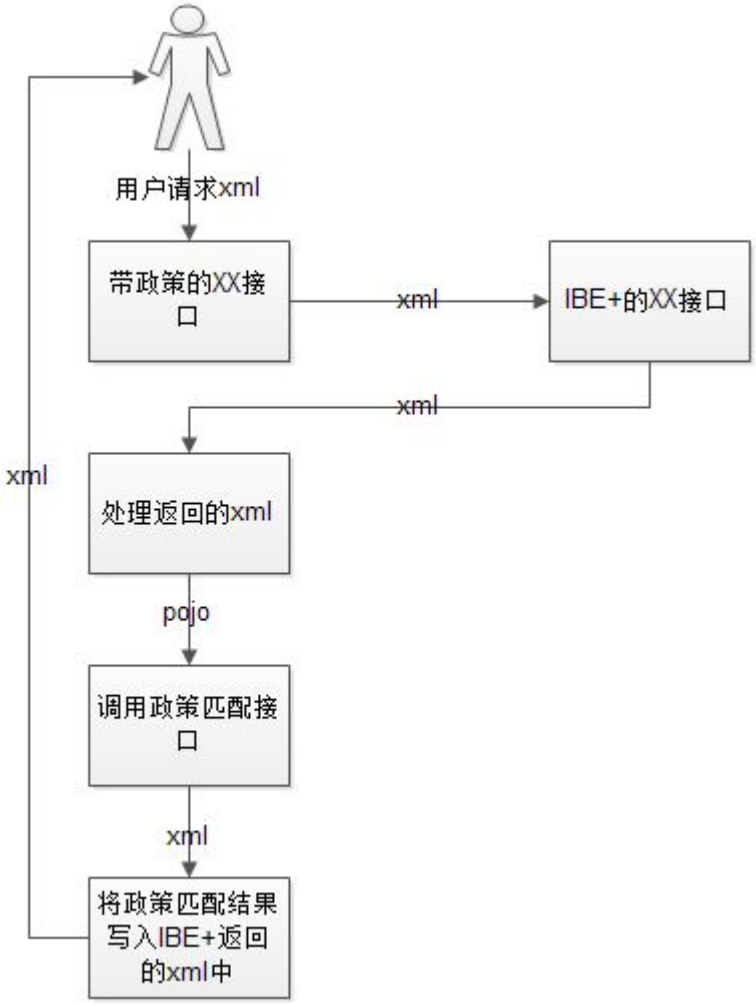
用户请求的 xml 具体见 xsd 文件。



上图为用户请求 xml 的一个例子，红框标识部分为必填内容。

3.6.3 模块的设计思路

根据需求，我们得出如下设计思路图：



上图中的 XX 接口在本节表示国际 pricing 接口。

简单的说就是用户调用我们国际 pricing 接口，我们根据用户请求的 xml 去调用 IBE+国际 pricing 接口，并根据其返回值调用政策匹配的接口，然后将政策匹配的结果写入到 IBE+返回的 xml 中，最后返回给用户。

3.6.4 模块的数据库设计

这个模块不涉及到数据库。

3.6.5 模块的限制条件

本模块对用户输入的 xml 必填标签验证如下：

一代 office 号

分销商

团队类型

如果其他必填标签没有填写，在请求 IBE+接口时，IBE+接口后报相应的异常，返回具有 Error 标签的 xml，我们验证如果返回的 xml 有 Error 标签，就直接返回给用户。

如果上述三个标签的某一个参数不填写，如分销商标签在请求 xml 中没有填写，则会报“分销商参数必填”的错误，并返回给用户。

3.6.6 模块的代码设计

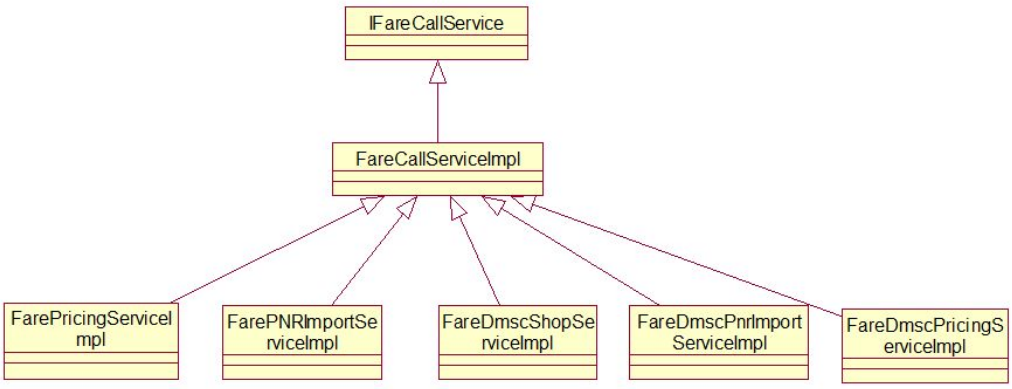
含政策的国际 pricing 接口是我们利用 webservice 对外发布的接口，当用户调用这个接口时，接口再调用我们内部系统进行的处理。

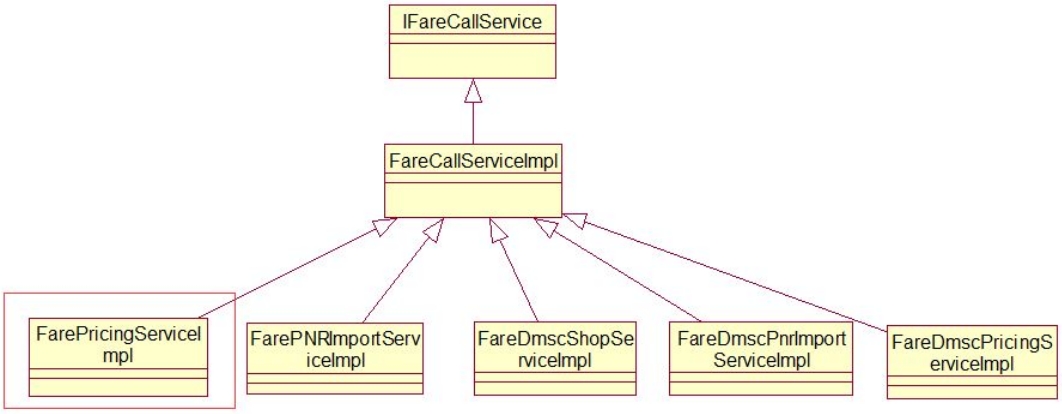
在我们系统内部，主要管理 webservice 接口调用的 API 接口为：IFareCallService，里面有如下 2 个方法：

String fareCallService(String xml)：本方法供 webservice 接口调用。

String fareCallService(String xml,boolean isdebug)：本方法供调试页面调用。

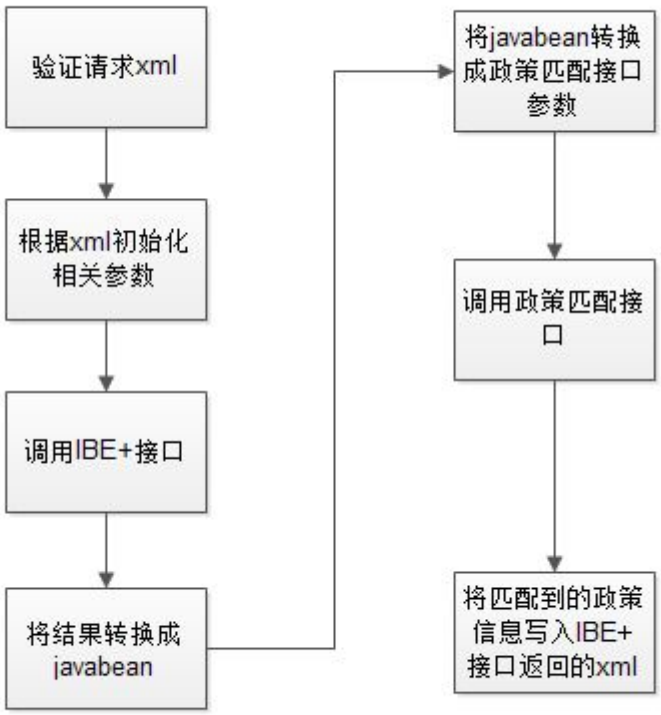
IFareCallService 接口的类族关系图如下：





针对含政策的国际 pricing 模块，主要应该关注 FarePricingServiceImpl 这个实现类。

在这里我们重点应该注意 FareCallServiceImpl 这个抽象类。在这个抽象类里 areCallService(String xml)这个方法内部调用了 fareCallService(String xml,boolean isdebug)这个方法。并且在 fareCallService(String xml,boolean isdebug)这个方法里定义了具体业务处理逻辑流程，处理逻辑流程如下图所示：



FarePricingServiceImpl 这个实现类继承了 FareCallServiceImpl 这个抽象类，并实现了其相关抽象方法。接下来我们针对上面的流程图作详细的介绍：

3.6.6.1 验证请求 xml

这个步骤是在 FareCallServiceImpl 抽象类中实现的，主要是根据需求验证用户请求的 xml 中是否含有 GroupType、Distributor、logkey 这几个标签，尤其是前 2 个标签属于必填标签，如果没有，则抛出相关异常，中断任务执行。

3.6.6.2 初始化相关参数

这个步骤在 FareCallServiceImpl 类中属于抽象方法，由子类实现，对于本模块，由 FarePricingServiceImpl 这个类实现。该功能主要是将请求 xml 中相关参数初始化成 javabean 对象(CallContext 对象)。

CallContext 对象主要包括：请求 IBE+的 xml、分销商信息、团队类型、一代 office 号、logkey、出票时间、出票航司、旅客类型和人数等属性。

将 xml 转换成 CallContext 对象的实现步骤如下：

1、初始化 IBE+请求 xml：将用户请求的 xml 中，去掉 GroupType、Distributor、logkey 这几个节点。（以防 IBE+具有相同节点）

2、解析请求 xml：得到请求中的旅客类型和旅客数量、出票航司、出票时间、一代 office 号，并将这些信息设置到 CallContext 对象中。

3、调用 InitBaseInfo 的静态方法 getBaseInfo 得到请求中的分销商、团队类型、logkey 等信息。

3.6.6.3 调用 IBE+接口

这个步骤主要是调用 IFareRemoteService 接口的 FarePricingCallServiceImpl 实现类的方法，去调用 IBE+接口，并根据获得接口返回的 xml。

3.6.6.4 将 IBE+结果 xml 转换成 javabean

1. 将 xml 通过 SerializeUtil 工具类 deserializeObject 这个方法将 xml 转换成 TSK_AirfarePrice 对象（这需要和 xsd 文件一致）。

这里需要注意的是：返回的 xml 可能和 xsd 文件不一致，需要特殊处理，即将 xml 中不能转换的标签如<sita:CouponDataInfos>这个标签去掉。

2.如果 TSK_AirfarePrice 对象不为空，则判断该对象下的 Errors 对象是否为空，如果不为空，表示 IBE+遇到异常，我们需要记录日志，并抛出 AppException 异常，异常内容为 IBE+返回的 xml。

3. 如果 TSK_AirfarePrice 对象为空，则直接抛出 AppException(AppMsg.SHOPPING_RESULT_ERROR)异常。

3.6.6.5 将 javabean 转换成政策匹配接口参数

这个步骤主要是调用 PricingRSParser 的 initPolicyMatchParameter 方法实现的，具体见代码。因为这个部分和 xsd 关系比较紧密，如果需要了解请先熟悉国际 pricing 的 xsd 文档，再对照代码进行阅读。

3.6.6.6 调用政策匹配接口

这个步骤主要是调用国际政策匹配接口，进行政策匹配，至于政策匹配的具体逻辑见国际政策匹配模块的说明。

3.6.6.7 将匹配到的政策信息写入 IBE+接口返回的 xml

最后是将政策匹配的结果 xml，和 IBE+返回的 xml 结果进行合并，即将政策匹配结果 xml 写入到 IBE+返回的 xml 中</ota:PricedItineraries>这个节点前。

3.6.7 模块的异常情况

本模块的异常情况主要是用户必填参数没有填写的情况下，会报相应的异常，在 WS 接口方法里，对异常进行了统一处理，并组织具有 Error 标签的 xml 返回给客户端。如果在调用 IBE+接口出现异常，则 IBE+直接返回的 xml 中已经具有了 error 标签，我们在解析的时候发现 error 标签，直接将此 xml 返回给用户。

3.6.8 模块尚未解决的问题

暂无

3.7 国际 PNR 导入模块详细设计

3.7.1 模块概述

国际 pnr 导入模块，是指用户调用我们对外发布的国际 pnr 导入接口时，我们根据用户请求的 PNR，调用 IBE+底层国际 pnr 导入接口，获得 PNR 运价信息，并根据运价信息调用我们自己的国际政策匹配接口，获得运价信息对应的政策信息，并将运价信息和政策信息返回给用户。至于如何进行政策匹配将在政策匹配模块进行介绍。

3.7.2 模块的需求描述

国际 pnr 导入模块，是指用户调用我们对外发布的国际 pnr 导入接口时，我们根据用户请求的 PNR，调用 IBE+底层国际 pnr 导入接口，获得 PNR 运价信息，并根据 PNR 运价信息调用我们自己的国际政策匹配接口，获得运价信息对应的政策信息，并将运价信息和政策信息返回给用户。

用户请求的 xml 具体见 xsd 文件。

3.7.3 模块的设计思路

和国际 pricing 类型。

3.7.4 模块的数据库设计

这个模块不涉及到数据库。

3.7.5 模块的限制条件

和国际 pricing 模块类似。

3.7.6 模块的代码设计

国际 PNR 导入接口在我们内部系统的处理和国际 pricing 类似，具体区别见 FarePNRImportServiceImpl 子类。

3.7.7 模块的异常情况

和国际 pricing 类似。

3.7.8 模块尚未解决的问题

暂无

3.8 国际 shopping 模块详细设计

3.8.1 模块概述

国际 shopping 模块，是指用户调用我们对外发布的国际 shopping 接口时，我们根据用户请求的 xml，调用 IBE+底层国际 shopping 接口，获得运价信息，并根据运价信息调用我们自己的国际政策匹配接口，获得运价信息对应的政策信息，并将运价信息和政策信息返回给用户。至于如何进行政策匹配将在政策匹配模块进行介绍。

3.8.2 模块的需求描述

国际 shopping 模块，是指用户调用我们对外发布的国际 shopping 接口时，我们根据用户请求的 xml，调用 IBE+底层国际 shopping 接口，获得运价信息，并根据运价信息调用我们自己的国际政策匹配接口，获得运价信息对应的政策信息，并将运价信息和政策信息返回给用户。

用户请求的 xml 具体见 xsd 文件。

3.8.3 模块的设计思路

和国际 pricing 类型。

3.8.4 模块的数据库设计

这个模块不涉及到数据库。

3.8.5 模块的限制条件

和国际 pricing 模块类似。

3.8.6 模块的代码设计

国际 shopping 接口在我们内部系统的处理和国际 pricing 类似，具体区别见 FareFlightShopServiceImpl 子类。

3.8.7 模块的异常情况

和国际 pricing 类似。

3.8.8 模块尚未解决的问题

暂无

3.9 国内政策维护模块详细设计

3.9.1 模块概述

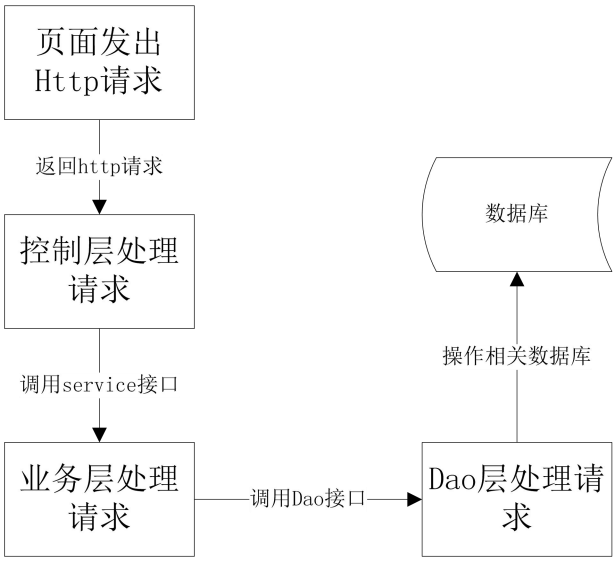
国内政策维护模块，是指用户可以通过模块随时增加政策、删除政策、修改政策、查询政策，达到对政策的日常维护。

3.9.2 模块的需求描述

国内政策维护模块，用户在需要增加、删除、修改、查询政策的时候，可以实现相应的操作政策。每个用户都只能操作自己的政策。

3.9.3 模块的设计思路

根据需求，我们得出如下设计思路图：



简单的说就是用户通过操作我们的系统,如添加政策时,录入政策信息，保存到我们系统的数据库信息中；查询时，即从数据库中取出信息，显示在系统界面上；删除时，我们并不是真正的删除政策信息，而是通过设置截止时间，假定删除一条政策，把数据依然存在数据库中，以备查阅。

3.9.4 模块的数据库设计

与数据库对应的表的字段 详情见 PolicyDmscPo.java 这个类

3.9.5 模块的限制条件

对于政策维护模块,分别有两个有效性验证，一是政策添加和修改页面的页面限制条件，我们必须限制有些必输项以及部分数据需要校验格式的正确性，具体情况如下表：

名称	类别	规格	备注
基础信息			
出票航空公司	文本框，必填	两字码，多个用斜线分隔，小写自动转大写。	举例：MU/FM 需要动态校验两字码是否存在
投放分销商	文本框，必填	自由文本，多个用斜线分隔	不校验，匹配时和传入内容比对
批发商	文本框，	自由文本，多个用斜线分隔	不校验
旅客信息			
旅客身份	文本框，必填	填写旅客类型三字码，同“shopping 请求”中的 PassengerTypeCode.默认为 ADT。	举例：ADT/CNN/INF
旅客人数下限	文本框，必填	数字（3），默认 1	
政策适用类型	文本框，必填	选项：散客，团队，散客/团队。默认：散客/团队	举例：散客/团队
地理位置信息			
行程类型	下拉框	选项：单程、往返、单程/往返，默认：单程/往返。	
出发机场/城市	文本框，必填	机场/城市三字码，多个用斜线分隔。	举例：PEK/TSN； 基础数据，需要校验是否存在
到达机场/城市	文本框，必填	机场/城市三字码，多个用斜线分隔。	基础数据，需要校验是否存在
日期要求			
去程旅行日期	文本框 必填	yyyymmdd，范围用-，多个斜线分隔。	2014091-20140930/2014908-20141231
回程旅行日期	文本框	同上 w	
出票日期	文本框 必填	必须是一个范围，YYYYMMDD-YYYYMMDD	2014091-20141231
更多限制条件			
去程仅限航班号	文本框	两字码+航班号。可以是单个航班或范围，多个用斜线分隔。	CA90-999/CA9999
去程排除航班号	文本框	同上	
回程仅限航班号	文本框	同上	
回程排除航班号	文本框	同上	
去程适用班期	复选框	选项：星期一~星期日，默认全选。	
回程适用班期	复选框	选项：星期一~星期日，默认全选。	
调价数额			
舱位 I	文本框 必填	填写舱位字母，可以多个，斜线分隔	举例：Y/H/B；也可以填 ALL，表示所有舱位。
返点 I	文本框	正数或负数，整数或一位小数，	举例：-2.5；（表示-2.5%）
费用 I	文本框	正数或负数，整数。	举例：-90；
舱位 II	文本框	同“舱位 I”	

表 2 国内政策信息表

二是政策查询页面时,我们需要限制查询的条件,只能根据那些条件来查询政策，具体情况如下表：

表 3 国内政策信息表

名称	类型	规格	备注
出票航司两字码	文本框，必填	数字/字母（2）	航空公司两字码，查询部区分大小写
投放分销商	文本框	无	投放分销商，查询不分大小写
出发机场/城市	文本框	三字码	
到达机场/城市	文本框	三字码	
形成类型	下拉框	选项：OW、RT、OW/RT，单选/不选	
生效日期	文本框	日期控件	手写格式为：YYYYMMDD

3.9.6 模块的代码设计

政策维护模块，外部用户主要是通过相关界面程序来操作我们系统的，用户只需要操作相应的按钮，即可操作我们系统。

在我们系统内部，主要是通过接受 web 页面传来的数据，然后通过 DB 连接操作数据库，从而达到对政策数据的基本维护。图类似国际。

在政策维护模块,我们主要功能实现是在 PolicyDmscServiceImpl.java 这个类里面，包括对政策的操作以及其他方法。在进行相关数据处理时,具体流程如下图：

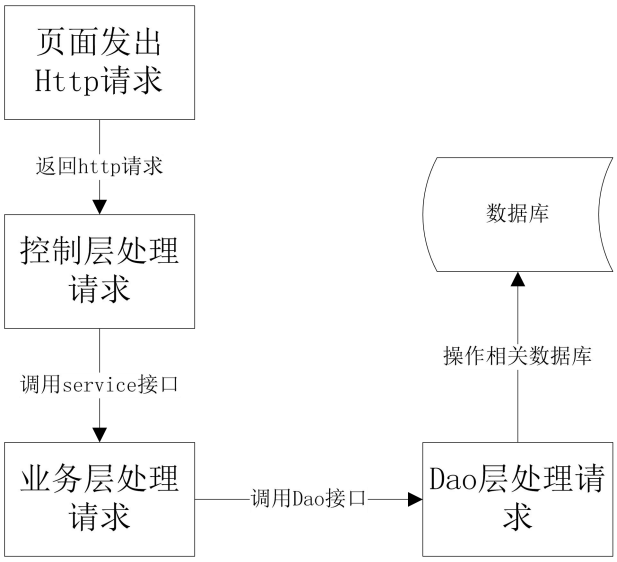


图 2 政策维护相关流程图

国内这块，还有个国内的工具类，PolicyDmscUtil.java，此类里面两个方法。passagerBoo(String passager)和 passagerTypeArr(String[] passagerArr);

passagerBoo 方法是为了 ajax 实现页面上旅客类型，不能填写 AD，CH，IN 大小写字样的做的返回结果。passagerTypeArr 方法，是当用户填写 ADT,CHD,INF 时，分别在该字段对应的数组字段中添加 AD，CH,IN。

3.9.6.1 页面发出 Http 请求

这个步骤是在用户操作 web 页面的时候发出的，用户通过点击页面上的按钮或者其他操作方式,向我们系统发出添加、修改、删除、查询等请求。

3.9.6.2 控制层处理相关请求

在这个步骤中,控制层主要负责将提交的请求分配给每个处理请求的方法,如添加请求，我们将调用 create 方法来处理，依次类推，在处理方法中我们会对传输的数据做一些格式或者验证处理，但我们不会对其进行逻辑处理。

3.9.6.3 业务层相关处理

这个步骤主要是对每个请求的逻辑进行处理的步骤。用户想要添加一个政策,我们把经过格式处理的数据，写入一些逻辑，如相关时间，相关人等等，最后调用 Dao 层接口处理数据。

3.9.6.4Dao 层处理

Dao 层主要是数据访问层，是访问数据的接口层,这里我们可以通过一些框架或者直接用 Sql 语句来操作数据库，因此,我们收到从业务层传入的相关请求，然后按照方法对数据库进行相应的操作。

3.9.6.5 数据库

这里主要是存放我们政策数据信息的。

3.9.7 模块的异常情况

本模块的异常情况主要有：

1、用户必填参数没有填写或者填写不符合规范的情况下，会直接在 web 页面提示用户。

2、相关逻辑异常,如果发现有相关逻辑的异常,我们会终止当前操作，并撤销已经操作的操作，并提示用户。

3.9.8 模块尚未解决的问题

暂无

3.10 国内政策批量导入详细设计

3.10.1 模块概述

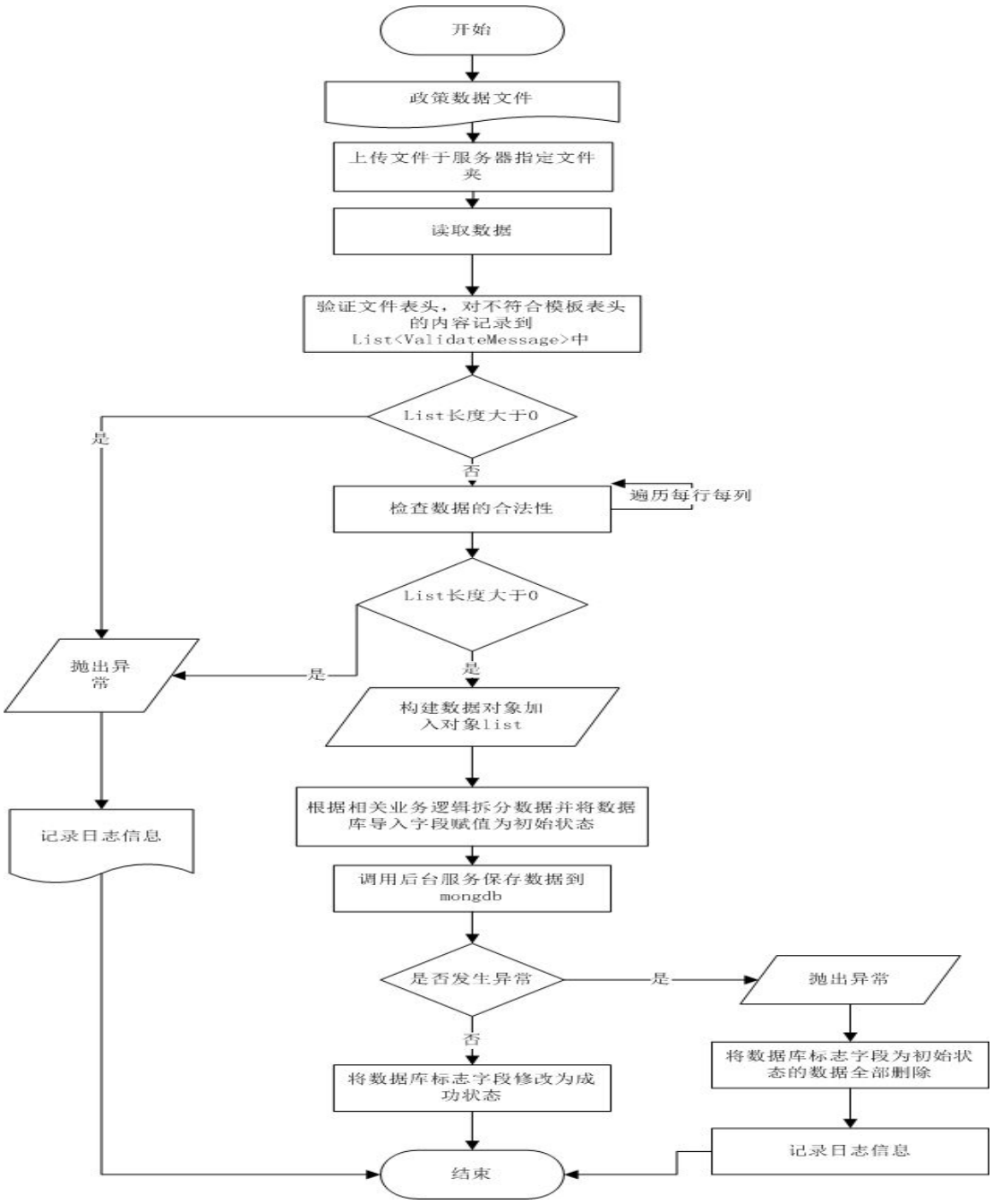
国内政策批量导入，主要是指用于根据系统指定的模板将大量的数据放入 excel 中，一次性导入到系统中，减少页面输入环节，提高政策添加效率。此模块主要涉及 excel 文件上传、数据格式的校验、入库等操作，另外系统在导入过程中实时将导入过程通过日志形式打印在前台页面，以供用户明确导入进程。

3.10.2 模块的需求描述

将一定格式的 excel 文档，在页面提交后，后台经过格式校验、数据解析、存入数据库，在此过程中前台实时打印处理日志。另外导入的数据必须和页面添加的政策数据保持一致。

3.10.3 模块的设计思路

根据需求，我们得出如下设计思路图：



根据上述图所示，我们的设计思路是：首先我们将用户上传的 excel 文件保存到服务器，接着解析服务器上的 excel 文档，在解析文档的过程中我们加入表头验证（保证用户 excel 必须和系统提供的一致，提高数据的完整性）、每一项的格式校验（保证数据的正确性），表在验证的过程中，如果出现异常或与模板的不一致时，将错误信息存入 ErrorQueue 的 MessageList 之中，当每一次大的验证完成之后，自动检测 ErrorQueue 的 MessageList 是否为空，如果时空，则继续，不然就抛出异常，打印错误日志给用户，如果校验没有错误，根据相关逻辑将数据在存入数据库之前进行包装，整合。最后将整合完成数据存入 mongodb 数据库，在此检测存储是否异常，如果有异常，回滚保存数据并打印错误日志给用户，如果

没有异常，修改标志字段，提示用户导入成功。

3.10.4 模块的数据库设计

这个模块不涉及到数据库设计，数据库设计由政策维护模块设计。

3.10.5 模块的限制条件

本模块对用户导入的 excel 有以下限制：

1 导入的模板必须在符合系统规定的格式，包括表头的名字也必须一致，也不能交换顺序。

2 导入的模板必须是以.xls 或.xlsx 的 excel 文件。

3 导入的数据需满足一下格式：

名称	类别	规格	备注
基础信息			
出票航空公司	文本框，必填	两字码，多个用斜线分隔，小写自动转大写。	举例：MU/FM 需要动态校验两字码是否存在
投放分销商	文本框，必填	自由文本，多个用斜线分隔	不校验，匹配时和传入内容比对
批发商	文本框，	自由文本，多个用斜线分隔	不校验
旅客信息			
旅客身份	文本框，必填	填写旅客类型三字码，同“shopping 请求”中的 PassengerTypeCode.默认为 ADT。	举例：ADT/CNN/INF
旅客人数下限	文本框，必填	数字（3），默认 1	
政策适用类型	文本框，必填	选项：散客，团队，散客/团队。默认：散客/团队	举例：散客/团队
地理位置信息			
行程类型	下拉框	选项：单程、往返、单程/往返，默认：单程/往返。	
出发机场/城市	文本框，必填	机场/城市三字码，多个用斜线分隔。	举例：PEK/TSN； 基础数据，需要校验是否存在
到达机场/城市	文本框，必填	机场/城市三字码，多个用斜线分隔。	基础数据，需要校验是否存在
日期要求			

去程旅行日期	文本框 必填	yyyymmdd，范围用-，多个斜线分隔。	2014091-20140930/2014908-20141231
回程旅行日期	文本框	同上	
出票日期	文本框 必填	必须是一个范围，YYYYMMDD-YYYYMMDD	2014091-20141231
更多限制条件			
去程仅限航班号	文本框	两字码+航班号。可以是单个航班或范围，多个用斜线分隔。	CA90-999/CA9999
去程排除航班号	文本框	同上	
回程仅限航班号	文本框	同上	
回程排除航班号	文本框	同上	
去程适用班期	复选框	选项：星期一~星期日，默认全选。	
回程适用班期	复选框	选项：星期一~星期日，默认全选。	
调价数额			
舱位 I	文本框 必填	填写舱位字母，可以多个，斜线分隔	举例：Y/H/B；也可以填 ALL，表示所有舱位。
返点 I	文本框	正数或负数，整数或一位小数，	举例：-2.5；（表示-2.5%）
费用 I	文本框	正数或负数，整数。	举例：-90；
舱位 II	文本框	同“舱位 I”	
返点 II	文本框	同“返点 I”	
费用 II	文本框	同“费用 I”	
舱位 III	文本框	同“舱位 I”	
返点 III	文本框	同“返点 I”	
费用 III	文本框	同“费用 I”	
混舱返点选择	下拉框	选项：1/2，较低，较高	
混舱费用选择	下拉框	选项：1/2，较低，较高	
代理费	文本框	正数，整数，2 整数。	区别于航空公司默认代理费时，可以填入。
销售配置	文本框	自由文本	
销售备注	文本域	字符串（250）	

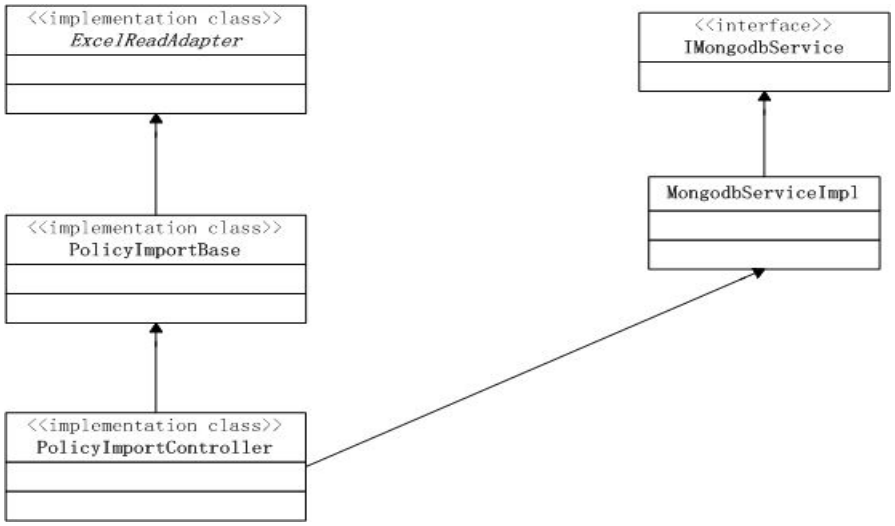
3.10.6 模块的代码设计

国内政策导入在设计时主要分成了 3 个模块：政策数据解析、校验、数据的校正、导入 mongodb。

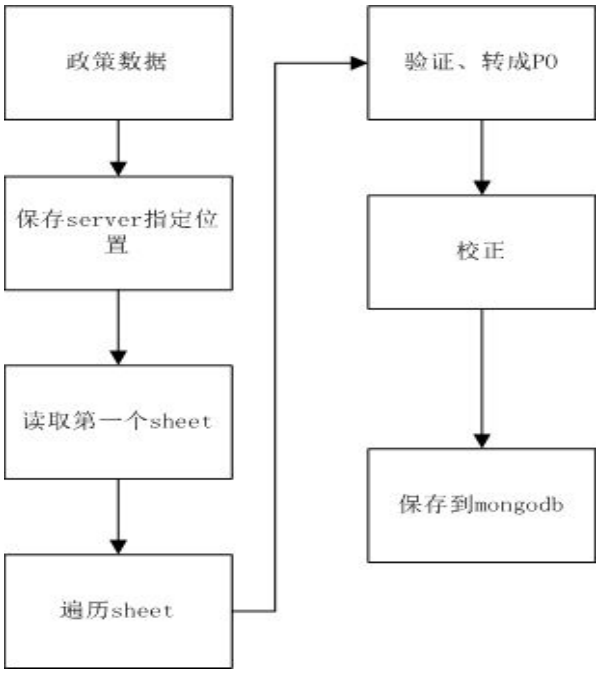
1 在政策数据的解析和校验模块，定义一个 excel 通用的抽象类 ExcelReadAdapter<T>类解析数据校验数据，并且将解析完成的数据加入政策对象 List 中，在具体的子类实现中定义 PolicyImportBase 来继承 ExcelReadAdapter<T>，具体的实现政策的验证、数据的校正。

2 在完成校验和数据的校正之后，调用 MongodbService 类完成对数据的批量导入。

下图为本模块的相关类的依赖关系。



导入的整个处理流程是，用户提交 excel 表格，服务器将用户提供的 excel 重新命名后保存到指定位置，然后读取解析 excel 的第一个 sheet，接着遍历 sheet 的每一行，每一列进行一下操作：获取数据->验证格式->装成对象属性，完成此操作之后校正数据，最后存入 mongodb。具体的如下图所示：



3.10.6.1 保存 excel 文件到指定的文件夹和读取 sheet

调用系统中工具类 FilesUtil 的 uploadExcel 方法完成 excel 的保存，并返回保存的路径，以供后续读取。如果保存有异常，直接中断响应，返回日志信息给用户。

调用 ExcelUtil 的 getSheetByExecl 的方法来读取 sheet。

3.10.6.2 遍历 sheet 和解析、校验数据

此模块的操作类是 PolicyImportBase，他继承了 ExcelReadAdapter，在 paserToPO 的方法中具体的实现的验证。在 analyticalData 方法中完成了数据的校正和初始化存入数据库的标志字段。

3.10.6.3 保存数据到 mongodb

这个模块主要是调用 MongodbServiceImpl 的 saveListObject 方法批量插入数据，完成之后更新标志字段，如果出现异常删除刚导入的数据。

3.10.7 模块的异常情况

导入模块对于系统级别的异常，在 Controller 层进程 try-catch，打印日志信息，给用户返回”系统异常“，在政策校验和校正中的异常会计入 ErrorQueue 的 messageList 中，返回给用户详细的提示。

3.10.8 模块尚未解决的问题

暂无

3.11 国内政策导出详细设计

3.11.1 模块概述

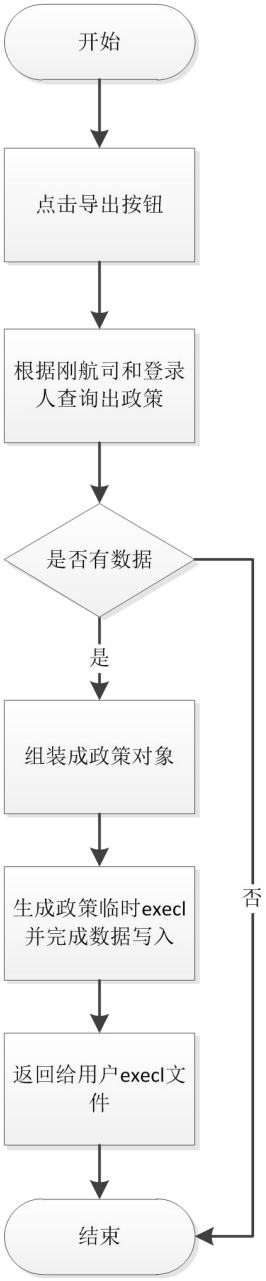
国内政策导出主要功能是根据航司以 execl 形式导出登录人的添加的政策。

3.11.2 模块的需求描述

政策维护人员进入政策导出页面，输入航空公司两字码，点击“开始导出”按钮。导出数据仅指当前政策库中所有有效的政策，已过有效期和未来有效的政策不能导出。待数据导出后，由浏览器弹出提示并保存 excel 文件。

3.11.3 模块的设计思路

根据需求，我们得出如下设计思路图：



根据上述图所示，用户点击导出按钮，系统根据条件查询出数据，组成成政策对象，写入到临时 execl 中，最后将 execl 文件传给用户。

3.11.4 模块的数据库设计

这个模块不涉及到数据库设计，数据库设计由政策维护模块设计。

3.11.5 模块的限制条件

无

3.11.6 模块的代码设计

本模块代码涉及以下主要类：

5、PolicyDmscImportController此类中的public String export(String airLine, Model model,HttpServletResponse response, HttpSession session)主要接受用户导出政策的请求，并查询政策数据，调用政策对象组装方法组装成对象，在调用execl写入方法写成execl提供并传输给用户。

6、ExcelUtil 类 中 的 policyDmscToXlsExcel(String modlePath,String outName,List<PolicyDmscPo> policyList) throws SysException生成一个execl临时文件并调用对象组装方法setPolicy(HSSFRow row, PolicyIntlPo policyIntlPo,HSSFCell cell, HSSFCellStyle style)来组装对象。

7、FilesUtil 类 中 的 exprotExcel(String filePath,String fileName, HttpServletResponse response)throws SysException 导出文件给用户。

3.11.7 模块的异常情况

捕获未知异常，并将信息返回页面。

3.11.8 模块尚未解决的问题

暂无

3.12 国内政策匹配模块详细设计

3.12.1 模块概述

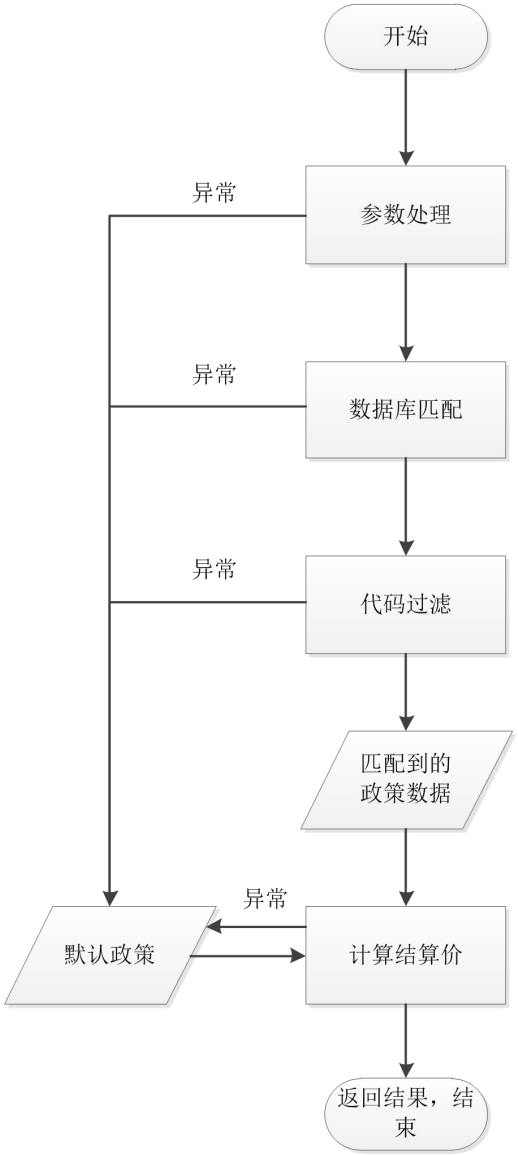
该模块通过传入的参数（参数来源于调用国内 shopping 接口、国内 pricing 接口和 pnr 导入接口后所返回信息中提取，主要包含运价信息、行程信息、乘客信息和基础信息等）与系统中的正政策进行匹配，最终通过匹配到的政策中的相关信息计算出每个运价信息的最终结算价，并返回给调用者。

3.12.2 模块的需求描述

用户调用“国内 shopping+政策”接口，或“国内 pricing+政策”接口，或“国内 PNR+政策”接口，之后先调用国内 shopping 接口，或国内 pricing 接口，或国内 PNR 计算接口，拿到计算结果后，再拼装成政策请求，进行政策匹配和结算价的计算，最终将政策结果和计算结果一起返回给用户。

3.12.3 模块的设计思路

根据需求，我们得出如下设计思路图：



简单的说就是调用政策匹配方法后，首先需要对参数做一些处理，然后对大部分条件进行数据库的匹配，然后再通过代码过滤方式将其它条件匹配上，再则用匹配到的政策计算出结算价，最终将结果返回。在上面的任意一步中遇到异常，则直接使用默认政策。

3.12.4 模块的数据库设计

与国内政策模块维护相同。

3.12.5 模块的限制条件

本模块由上级模块调用，需要传入 FareDmscRequest 对象参数，包含：office 号（一代信息）、分销商（二代信息）、出票日期、团散类型、日志 key、是否调试、航段信息、乘

客信息和运价信息，其中航段信息、乘客信息和运价信息可能有多个（比如：shopping 结果一次可能返回多个）。

输入参数	参数说明	是否必填	举例	备注
分销商信息	分销商集合名称，和政策数据中一致	是	Sele1	
出票时间	如果是历史政策匹配，需传入历史出票时间。默认为当前时刻。	否	2014-3-11 19:00	Shopping 和 Pricing 只传一个出票时间。当出票日期早于当前日期时，认为是历史匹配。
拼散包团	标识查询政策是查散客还是团队，默认散客。	否	散客	
旅客类型代码	旅客类型三字码	是	ADT	同 shopping 或 pricing 接口里的旅客类型代码
旅客人数	实际行程中每个旅客类型的人数，默认为 1。	是	2	同 shopping 或 pricing 接口里的旅客数量
序号	航段标识	是	1、2、3、4...	
出发机场	机场三字码	是	PEK	
到达机场	机场三字码	是	SHA	
出发日期	起飞的日期	是	2014-4-18	
出发时间	起飞的时刻	是	11:20	
到达日期	降落的日期	是	2014-4-18	
到达时间	降落的时刻	是	19:15	
市场方	市场方航空公司两字代码	是	CA	
承运方	承运方航空公司两字代码，不填则认为与市场方一致。	否	MU	
航班号	航班号	是	CA991	航空公司+航班号
舱位	舱位代码	是	B	
出票航空公司	TicketingAirline，两字码。	是	CA	
票面价	票面价格，不含税，	是	2600	

	BaseFare。			
税款	税费总金额。	是	1249	
Q 值	附加费用。	否	40	
代理费	代理费率	否	3	

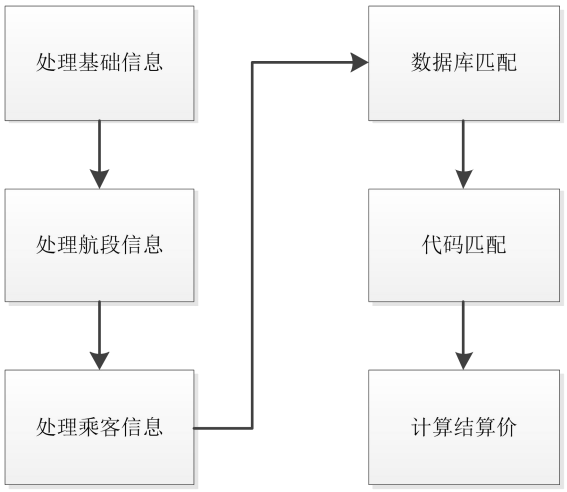
3.12.6 模块的代码设计

国内政策匹配接口 IPolicyDmscMatchService 中，只包含一个方法

String fareQuery(FareDmscRequest fareRequest)，该方法接受上层方法调用，最终返回拼装好的 xml。

其中 SinglePolicyDmscMacthService 作为贯穿整个匹配的一条线，其中的 public void doSingle(SingleInfoRS singleInfoRS) 方法作为入口。

处理逻辑流程如下图所示：



接下来我们针对上面的流程图作详细的介绍：

3.12.6.1 处理基础信息

private void doBaseInfoLogic(SingleSqlDmscRS singleSqlDmscRS, SingleInfoRS singleInfoRS) 该方法主要是设置一些基础信息，和一代信息处理；

3.12.6.2 处理航段信息

private void doFlightLogic(SingleSqlDmscRS singleSqlDmscRS, SingleInfoRS

singleInfoRS) throws AppException，通过该方法可得到行程类型，行程起始点,。同时如果航段数大于 2，则抛出应用异常，使其走默认政策。

3.12.6.3 处理乘客信息

private void initPassengerParams(SingleSqlDmscRS singleSqlDmscRS, SingleInfoRS singleInfoRS)

该方法主要是将乘客人数和乘客类型设置到参数中用于代码匹配。

3.12.6.4 数据库匹配

数据库匹配主要是用到 MatchPolicyDmscNoSqlService 类。

首先是通过 InitMatchPolicyDmscNoSql 类的 initMatchingFareSql（）方法获得从 mongodb 数据库匹配国内政策的查询条件；

然后，通过条件查询缓存中是否已经有该查询条件的结果，通过这个 private List<DBObject> getResultsFromCache(SingleSqlDmscRS singleSqlDmscRS, BasicDBObject queryObjects)方法实现；

如果缓存中没有找到结果，则用查询条件到数据库中查找，并将结果存放到缓存中。

最后得到结果后，将数据返回，如果通过条件得到的结果为空，则抛出应用异常，使程序走默认政策。

3.12.6.5 代码匹配

这个步骤主要是将数据库匹配到的政策再用剩余的条件再过滤一次，最终得到符合条件的多条政策（也可能过滤后就没有政策了）。

代码匹配在 MatchPolicyDmscNoSqlService 的 public void matchingCodeFare(SingleCodeDmscRS singleCodeDmscRS) 方法中实现。其核心代码主要在 PostValidateDmscContext 验证上下文和 IPolicyDmscPostValidate 验证接口的具体实现类。

在系统启动时，将所有验证规则加载到上下文中（PostValidateDmscContext 的 public void init() 方法），在代码匹配时调用 PostValidateDmscContext 的 public boolean handlePostValidate(SingleCodeDmscRS singleCodeDmscRS)的方法，进行校验。其中校验规则包含：1）去程旅行日期，2）回程旅行日期，3）.去程仅限航班号，4）去程排除航班号，5）

回程仅限航班号，6）回程排除航班号，7）舱位。

3.12.6.6 计算结算价

计算结算价就是通过政策中的相关信息（代理费、返点、其它费用）和接口传入的运价信息（票面价、税费）计算出最终结算价。

计算公式：结算价=票价*（1-代理费）（1-返点）+税款+费用。

其中默认政策的代理费默认为：3%

PriceCalcService 是计算价格的类，public void calcPrice(CalcPriceRs cabinPriceRs) 方法是计算价格的入口，如果有多条政策则会循环调用 private void calcPrice(TicketPriceInfoDmsc ticketPriceInfoDmsc,PolicyDmscResult policyDmscResult) throws AppException 方法继续计算，如果在计算过程中 由于有些原因 最终结算价 小于了 0 则会抛出应用异常 走默认政策计算。如果没有政策，则直接走 DefaultPolicyDmscHandler 的 public void doDefaultPrice(CalcPriceRs cabinPriceRs, List<PolicyDmscResult> policyDmscResults) 方法计算默认政策的结算价。

在计算最终结算价之前需要调用 private void cabinMatchPolicy(DBObject policy, String cabins, PolicyDmscResult policyDmscResult)方法根据舱位计算出返点信息。

3.12.6.7 结果返回

我们再回到 IPolicyDmscMatchService 的实现类 PolicyDmscMatchServiceImpl，关注 private String covertToXml(FareDmscResponse response)方法。该方法就是讲上面的计算出来的结果，转换为 xml。

最终的结构如下所示：

```
<PolicyBindings>
    <Policys>
        <Policy>
            <Seq>1</Seq>
```

```
            <Content>[{},{}]</Content>
        </Policy>
        <Policy>
            <Seq>2</Seq>
            <Content>[{},{}]</Content>
        </Policy>
    </Policys>
</PolicyBindings>
```

Policys 节点中是接口出入的多个运价信息，Content 节点中是一个 json 数组字符串，里面包含了匹配到的政策信息和对应的结算价信息等。

3.12.7 模块的异常情况

本模块异常有应用异常、系统异常和运行时异常，对应前面两种异常中已经处理，直接走默认政策，对于运行时异常，上级调用则会进行捕获，返回错误信息给用户。

3.12.8 模块尚未解决的问题

暂无

3.13 国内 pricing 模块详细设计

3.13.1 模块概述

国内 pricing 模块，是指用户调用我们对外发布的国内 pricing 接口时，我们根据用户请求的航段信息，调用 IBE+底层国内 pricing 接口，获得航段运价信息，并根据运价信息调用我们自己的国内政策匹配接口，获得运价信息对应的政策信息，并将运价信息和政策信息返回给用户。至于如何进行政策匹配将在国内政策匹配模块进行介绍。

3.13.2 模块的需求描述

国内 pricing 模块，是指用户调用我们对外发布的国内 pricing 接口时，我们根据用户请求的航段，调用 IBE+底层国内 pricing 接口，获得航段运价信息，并根据运价信息调用我们自己的国内政策匹配接口，获得运价信息对应的政策信息，并将运价信息和政策信息返回给用户。

用户请求的 xml 具体见 xsd 文件。

3.13.3 模块的设计思路

和国际 pricing 模块类似。

3.13.4 模块的数据库设计

这个模块不涉及到数据库。

3.13.5 模块的限制条件

和国际 pricing 模块类似。

3.13.6 模块的代码设计

含政策的国内 pricing 接口是我们利用 webservice 对外发布的接口，当用户调用这个接口时，接口再调用我们内部系统进行的处理。

国内 pricing 接口在我们内部系统的处理和国际 pricing 类似，具体区别见 FareDmscPricingServiceImpl 子类。

3.13.7 模块的异常情况

和国际 pricing 类似。

3.13.8 模块尚未解决的问题

暂无

3.14 国内 shopping 模块详细设计

3.14.1 模块概述

国内 shopping 模块，是指用户调用我们对外发布的国内 shopping 接口时，我们根据用户请求的 xml，调用 IBE+底层国内 shopping 接口，获得运价信息，并根据运价信息调用我们自己的国内政策匹配接口，获得运价信息对应的政策信息，并将运价信息和政策信息返回给用户。至于如何进行政策匹配将在国内政策匹配模块进行介绍。

3.14.2 模块的需求描述

国内 shopping 模块，是指用户调用我们对外发布的国内 shopping 接口时，我们根据用户请求的 xml，调用 IBE+底层国内 shopping 接口，获得运价信息，并根据运价信息调用我们自己的国内政策匹配接口，获得运价信息对应的政策信息，并将运价信息和政策信息返回给用户。

用户请求的 xml 具体见 xsd 文件。

3.14.3 模块的设计思路

和国际 pricing 模块类似。

3.14.4 模块的数据库设计

这个模块不涉及到数据库。

3.14.5 模块的限制条件

和国际 pricing 模块类似。

3.14.6 模块的代码设计

国内 shopping 接口在我们内部系统的处理和国际 pricing 类似，具体区别见 FareDmscShopServiceImpl 子类。

3.14.7 模块的异常情况

和国际 pricing 类似。

3.14.8 模块尚未解决的问题

暂无

3.15 缓存设计

3.15.1 模块概述

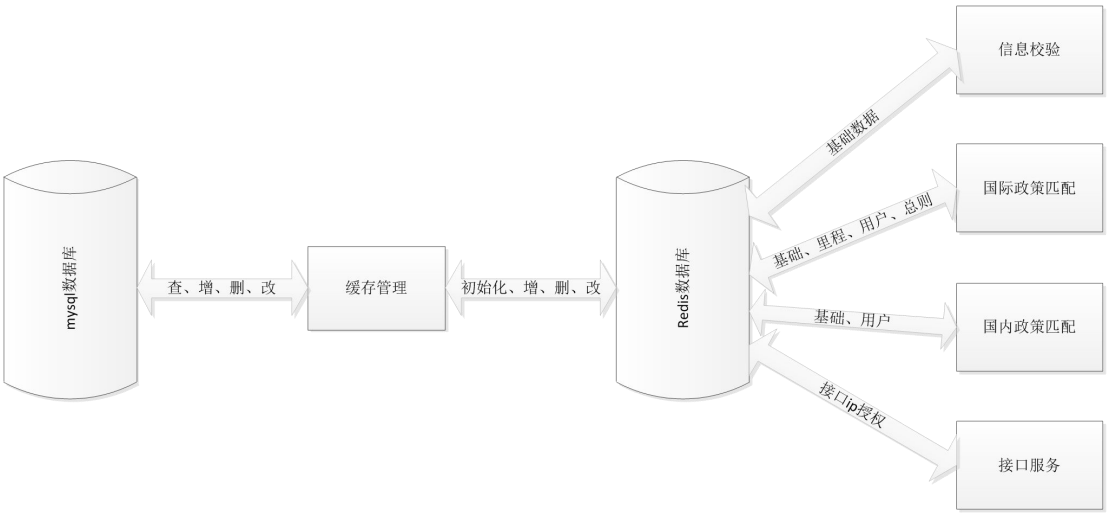
这里说的缓存设计指的是系统中使用 redis 缓存的地方，使用 redis 缓存的目的不言而喻，就是加快数据的查询速度。目前系统中所有存放在 mysql 数据库中的数据都加入到了 redis 缓存中。

3.15.2 模块的需求描述

该系统 mysql 数据库中存放的数据有用户数据、总则数据、代理费数据、接口 ip 授权数据、里程数据、基础数据（机场城市区域等），这些数据在匹配时可能会频繁的调用，为了提高匹配的效率，因此将数据加载到了 redis 缓存中。

3.15.3 模块的设计思路

根据需求，我们得出如下设计思路图：



简单的说就是缓存管理模块将 mysql 的数据按照一定的数据结构存储到 redis 中，然后并对外提供相关操作进行缓存的查询和更新。

3.15.4 模块的数据库设计

该模块的数据库设计请查考其它模块。

3.15.5 模块的限制条件

无。

3.15.6 模块的代码设计

缓存实现的过程是①初始化（将数据库中已有的数据初始化到 redis 中，在初始化前，先清空 redis 中的缓存）；②在对源数据进行增加、修改和删除时也对应改变 reids 中的数据；③缓存对外服务）。

初始化一般发生在系统启动时，也有时间触发（如：每天初始一次、或者页面点击初始化）。

下面将通过上面 3 点结合对应的基础数据进行说明。

3.15.6.1 代理费

代理费目前是是事先将个航司的代理费导入到 mysql 数据库中，系统只提供查询。在系统启动时调用 IAgencyFeeCacheService 接口的 init()方法将数据初始化到 redis 中，通过 AgencyFeeVo queryByCode(String airCode)方法查询缓存中的代理费实体；

3.15.6.2 接口 IP 授权

接口 IP 授权是针对用户调用接口时设置的 ip 白名单，只有白名单内包含的 ip 才能访问接口，否则向用户返回错误提示信息。

IIpAuthorizedDataCacheService 接口的 void initIpAuthorizedInfo()就是初始化化缓存；

boolean checkIpAuthorized(String ip)方法是判断传入的 ip 和缓存中的 ip 进行匹配判断是不是合法的；

void addIpAuthorized(String ip, String authorized);该方法是增加一个 ip，并设置是属于白名单还是黑名单；

3.15.6.3 基础数据缓存

基础数据缓存完成的任务是把航空公司和机场、城市、国际及区域等数据缓存。

其中航空公司缓存是按 map 形式（二字码-航司实体）形式存储；

城市信息缓存是按 map 形式（三字码-实体信息 FundamentalDataInfo）形式存储；

机场信息缓存是按 map 形式（三字码-实体信息 FundamentalDataInfo）形式存储；

所有基础信息缓存，是将所有的信息的机场、城市、省份、国家、大区、子区等信息存放到 set 中,其中城市、机场用 code 存储，城市以上级别使用对应汉子存储。

上面的 FundamentalDataInfo 主要包含的属性有：

城市三字码、城市中文名、机场三字码、机场中文名、机场中文简称、省份、国家、子区编码、子区中文名、子区英文名、大区、州。

3.15.6.4 里程缓存

该缓存设计在本文档的其它模块有详细说明；

3.15.6.5 总则缓存

首先是将 mysql 数据库中生效的总则数据全部查询出来，然后再通过 map 的形式存放到 redis 缓存中。其中 key 是：用户名+出票航司，value 是对应的总则实体；是实现方式是 IGeneralDataCacheService 接口的 init()方式；

当正常匹配时调用 public GeneralVo findByUserAndAirline(Long id, String airline)方法查找总则信息，其中 id 就是 office 对用的用户 id，airline 就是出票航司；

当总则维护时，增、删、改是需要调用 public void add(GeneralVo generalVo) 和 public void delete(GeneralVo generalVo)两个方法来更新缓存；

3.15.6.5 用户缓存

用户的缓存的设计稍微要复杂点，但仍然是以 map 形式的键值对形式存储，

其中 key 是 office 号，value 是一个用户实体；

由于系统中一个用户可以对于多个 office 号，因此在缓存设计时，需要将 office 分割开，也就是一个 office 号对应一个 user，多个 office 号可能对应相同的 user。例如：

用户表：

U1---->O1/O2

U2--->O3

则存储到缓存中：

O1--->U1

O2--->U1

O3--->U2

注意的是，每个用户之间的 Office 号不能有相同的。

3.15.7 模块的异常情况

缓存初始化时，不对外抛出异常，全部异常捕获，如发生异常记录日志即可。在调用缓存查询时，如遇到异常，将会抛出运行时异常，让上层调用者进行处理。

3.15.8 模块尚未解决的问题

用户模块对用户名和 office 号的唯一性校验，还未实现。

3.16 里程缓存模块

3.16.1 模块概述

里程缓存模块主要运价计算的基础信息，数据主要缓存在 redis 中，方便其他功能快速查询 OD 点的里程数。

3.16.2 模块的需求描述

里程缓存模块主要将 txt 文件中的里程数据通过代码方式将数据读入到数据中，再将数据从数据库中存入 redis 中，此数据是一次性保存在 redis 中，由于后续没有更改数据的可能性不大，未做修改模块，存入数据的关键字未出发+到达地，如 pek 到 sha，为 peksha，所存 value 为在数据库中取到的值放入 list 在放入缓存。

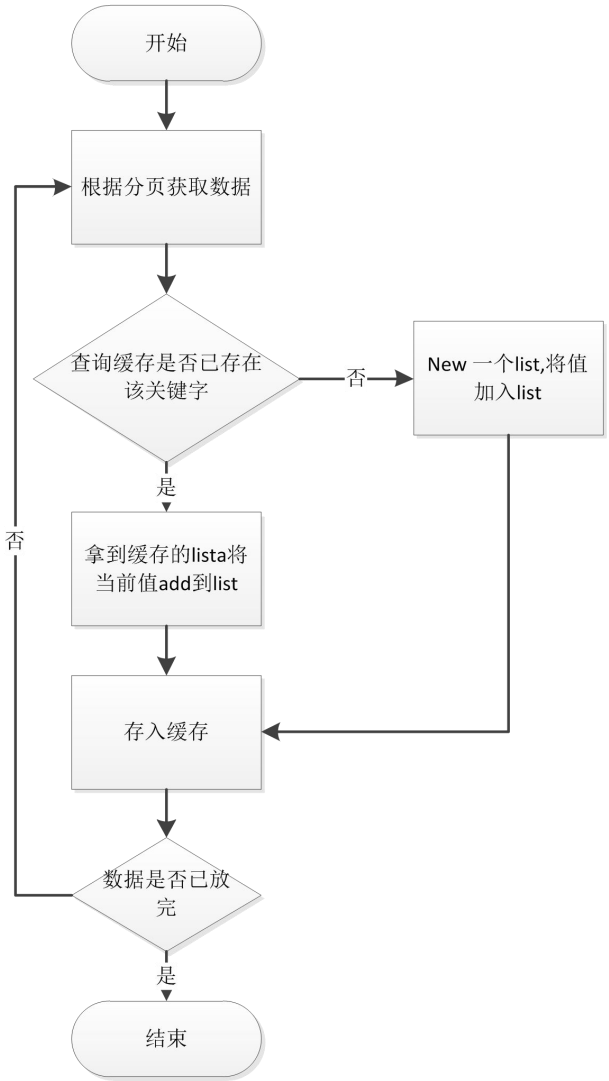
里程数据的读出，由于里程数据读出有两种 mileagempm 和 mileagetpm,以下称 mpm 和 tpm。

（1）当取 mpm 时，先去查询 mpm 数据，如果旅行方向无值有多条记录则反会最小值。如果查询参数旅行方向有值，则按旅行方向取值。如果 mpm 没有值则查询 tpm 数据，查询规则仍然像查询 mpm 的值得规则一样，如果获得值后对 tpm 的值乘以 1.2 为 mpm 的值。

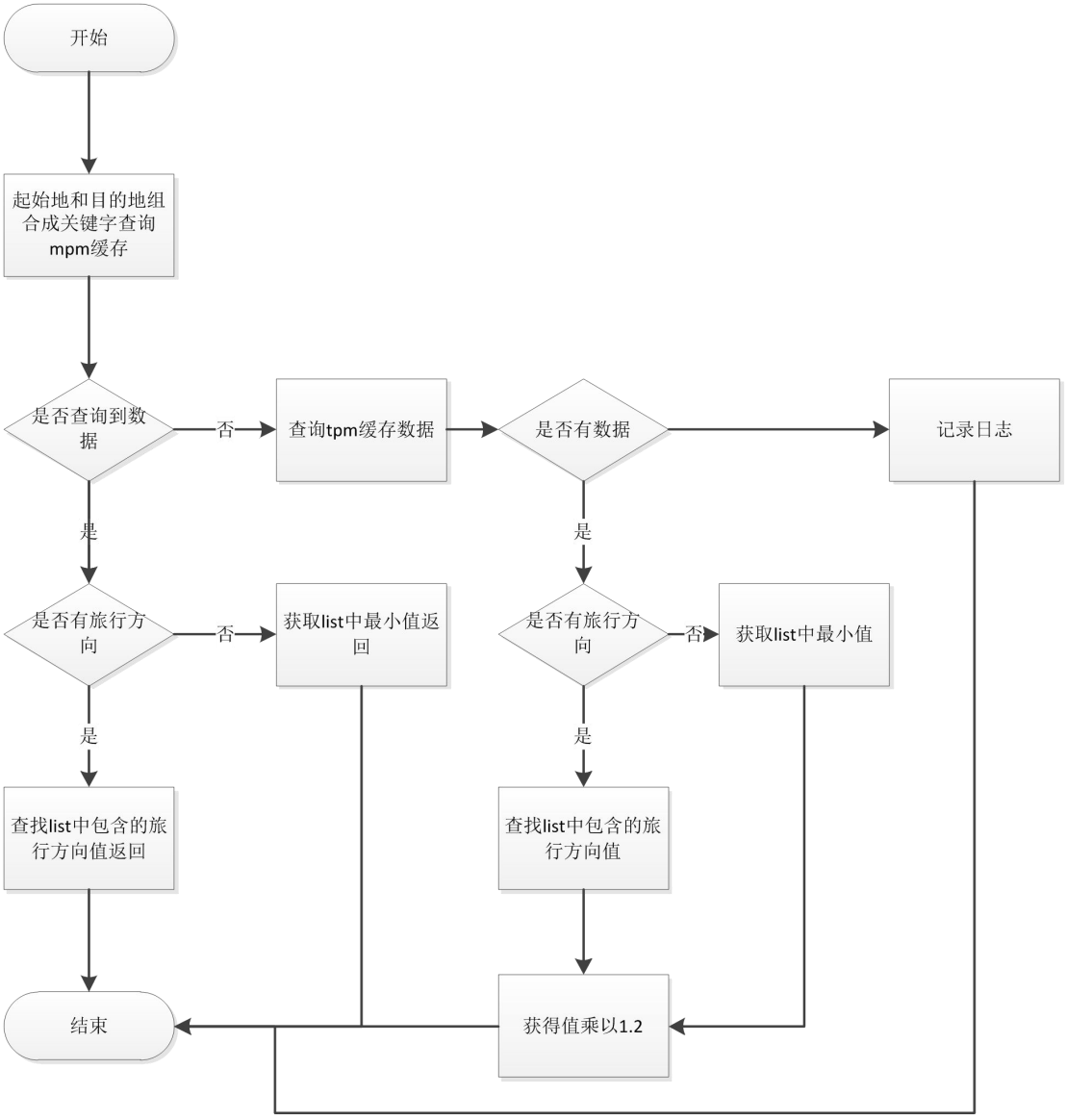
（2）当取 tpm 时，先去查询 tpm 数据，如果旅行方向无值有多条记录则反会最小值。如果查询参数旅行方向有值，则按旅行方向取值。如果 tpm 没有值则查询 mpm 数据，查询规则仍然像查询 tpm 的值得规则一样，如果获得值后对 mpm 的值除以 1.2 为 tpm 的值。

3.16.3 模块的设计思路

根据需求，存放缓存如下设计思路图：



获取缓存数据设计思路如下图：



3.16.4 模块的数据库设计

表名： tb_mileage_mpm				
序号	列名	数据类型	是否为空	列备注
1	id	int	NO	
2	start_number_code	varchar	YES	起点数字代码
3	end_number_code	varchar	YES	终点数字代码
4	end_city_name	varchar	YES	终点城市名
5	end_country_code	varchar	YES	终点国家代码
6	travel_direction	varchar	YES	旅行方向

7	shortest_distance_running	int	YES	SOM（最短运行距离）
8	maximum_allowed_stroke	int	YES	MPM（最大允许里程）
9	start_area_code	varchar	YES	起点区域代码
10	end_area_code	varchar	YES	终点区域代码
11	segment_code	varchar	YES	航段码
12	start_city_name	varchar	YES	起点城市名
13	start_country_code	varchar	YES	起点国家代码
14	start_air_code	varchar	YES	起点三字码
15	end_air_code	varchar	YES	终点三字码
16	is_min_mileage	varchar	YES	是否是最小里程

表名： tb_mileage_tpm

序号	列名	数据类型	是否为空	列备注
1	id	int	NO	
2	start_airport_type	varchar	YES	起点 *表示多机场，空表示单机场，#表示弃用
3	start_city_name	varchar	YES	起点城市名
4	start_country_code	varchar	YES	起点国家代码
5	start_air_code	varchar	YES	起点三字码
6	start_number_code	varchar	YES	起点数字代码
7	end_airport_type	varchar	YES	起点 *表示多机场，空表示单机场，#表示弃用
8	end_city_name	varchar	YES	终点城市名
9	end_country_code	varchar	YES	终点国家代码
10	tpm_mileage	int	YES	TPM 里程
11	travel_direction	varchar	YES	旅行方向

12	**mileage_type**	**varchar**	**YES**	**空表示直达里程，+表示构建里程**
13	**end_air_code**	**varchar**	**YES**	**终点三字码**
14	**end_number_code**	**varchar**	**YES**	**终点数字代码**
15	**start_area_code**	**varchar**	**YES**	**起点区域代码**
16	**end_area_code**	**varchar**	**YES**	**终点区域代码**

3.16.5 模块的限制条件

本模块无限制。

3.16.6 模块的代码设计

本模块主要业务逻辑在MileageDataCacheServiceImpl接口中，主要有提供了四个接口如下：

(1)void init()接口初始化里程数据mpm和tpm到缓存中。此过程时间较长。

(2)void initMileageMpmInfo(int page, int pageSize, Boolean flage)接口手动初始化mpm到里程中，主要逻辑是按照分页来从某一页重新开放初始化缓存数据。

(3)MileageMpmInfo getMileageMpmInfo(String startAirCode, String endAirCode,String travelDirection)接口从缓存中查询mpm数据。

(4)MileageTpmInfo getMileageTpmInfo(String startAirCode, String endAirCode,String travelDirection)接口从缓存中查询tpm数据。

3.16.7 模块的异常情况

本模块的异常情况主要是reids连接失败会抛出异常，或者redis存取数据超时抛出异常。发生此种异常会导致初始化缓存中断，而是里程本应该完整的数据导致只进行一半。

3.16.8 模块尚未解决的问题

暂无

3.17 接口服务发布模块详细设计

3.17.1 模块概述

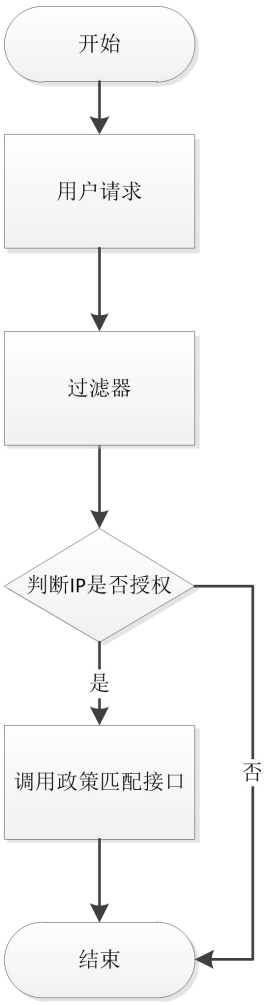
接口服务发布主要用于外部用户来运价系统进行政策匹配。主要技术使用了webservice 进行对外发布服务。接口服务提供国际 shopping、pnr、pricing 和国内shopping、pnr 和 pricing 六个对外接口，分别是不同的 maven 模块。以下以国际pnr 导入为列。

3.17.2 模块的需求描述

接口服务发布模块主要获取用户的请求分别调用各接口进行政策匹配然后返回结果给用户。在用户调用接口的时需要对用户的 IP 进行校验。

3.17.3 模块的设计思路

根据需求，得出如下设计思路图：



3.17.4 模块的数据库设计

这个模块不涉及到数据库。

3.17.5 模块的限制条件

本模块对用户的 IP 进行校验，如果 IP 未授权则不允许进行政策匹配。

3.17.6 模块的代码设计

3、主要配置文件 applicationContext-ws.xml，该文件主要配置接口发布的名称和接口注入。

4、拦截器 WebServiceAspect.java，该类主要对用户 IP 进行校验，接口异常的捕获后进行异常封装成 xml。此类以后可以扩展增加用户的其他信息验证。

5、PnrPolicyWS 实现了 IPnrPolicyWS 接口，次接口为对外发布的服务的接口，此类中使用了 IFareCallService 接口，通过注入不通的接口的实现来完成 pnr 或者 pricing 等接口的政策匹配。

3.17.6.1 验证请求 xml

这个步骤是在 FareCallServiceImpl 抽象类中实现的，主要是根据需求验证用户请求的 xml 中是否含有 GroupType、Distributor、logkey 这几个标签，尤其是前 2 个标签属于必填标签，如果没有，则抛出相关异常，中断任务执行。

3.17.6.2 初始化相关数据

本模块中重要的数据 IP 需加载到 redis 中，否则用户不能正常调用接口。

3.17.6.3 调用政策匹配接口

这个步骤主要是调用国际政策匹配接口，进行政策匹配，至于政策匹配的具体逻辑见国际政策匹配模块的说明。

3.17.6.4 将匹配到的政策信息的 xml 返回

最后是将政策匹配的结果 xml 返回给用户。

3.17.7 模块的异常情况

本模块的拦截会捕获政策匹配接口抛出来的异常，进行异常的封装，以 xml 的形式返回给用户。

3.17.8 模块尚未解决的问题

暂无