**第2章-保险基础数据-设计与开发**

**学习目标**

* 设计开发基础数据中的保障项、系数项、筛选项、保险分类功能

**1、基础数据介绍**

保险-基础数据。通过基础数据功能的学习我们主要解决下列问题：

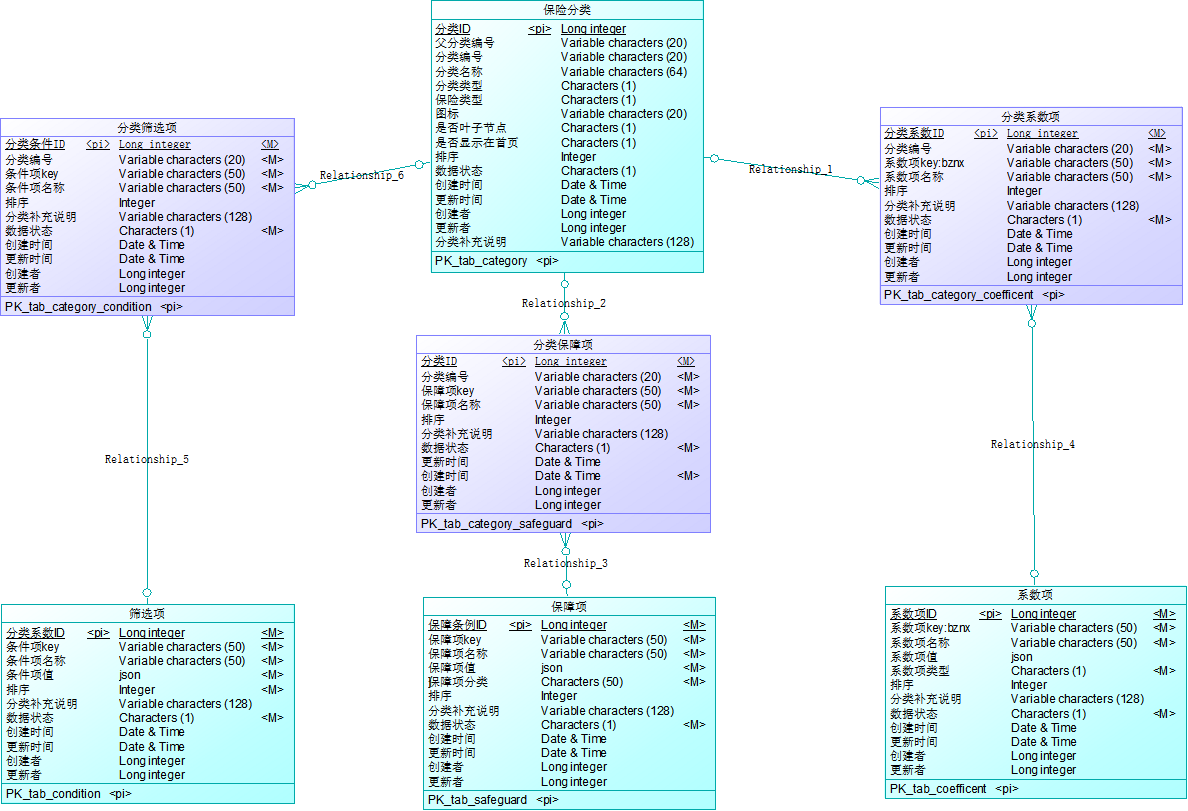
* 什么是基础数据，包含哪些，分别有什么作用？
* 基础数据的表结构如何进行设计？
* 分类与保障项、系数项、筛选项有什么关系？



上图是用户端功能截图，从中我们提炼并定义下列4个名词：

* **保障项：**通常是指在购买保险时所包含的具体风险或事件，以及在这些风险或事件发生时，保险公司愿意提供的经济赔偿或保障。不同类型的保险产品涵盖不同的保障项
* **系数项：**通常是指一系列用于计算保险费率的因子或系数。这些因子可以包括被保险人的年龄、性别、健康状况、驾驶记录、保险类型、地理位置等，它们会影响最终的保险费率计算。
* **筛选项：**通常是指一系列因素和选项，需要根据你的具体需求和情况来考虑，来选择自己需要的保险服务。
* **分类项：**通常是指保险根据不同的标准和覆盖范围进行的分类，不同的分类涵盖的方面不同，此处我们主要讨论下列几类保险：医疗、重疾、意外、养老金、旅游等分类；[附录-保险分类](https://j1wtmv7ajj.feishu.cn/wiki/Au95wpmubiP9emkDFr7coayFndb?from=from_copylink)

这几个数据结合目前四方保险数据库中的表实际情况而梳理的概念模型为：



从上述概念模型可以看出；分类与三类基础数据都有关系；与它们的关系为：

* 一个分类下可以有多个保障项，一个保障项也可以对应不同的分类；

|  |
| --- |
| 分类与保障项的关系为：多对多 |

* 一个分类下可以有多个系数项，一个系数项也可以对应不同的分类；

|  |
| --- |
| 分类与系数项的关系为：多对多 |

* 一个分类下可以有多个筛选项，一个筛选项也可以对应不同的分类；

|  |
| --- |
| 分类与筛选项的关系为：多对多 |

概览看起来比较多；我们逐个来实现，一个个分析；以后实际的开发也都将是如此，先一个业务分析，完成之后再做另外的。

**2、保障项-保险保障范围**

**2.1、需求分析**

由于项目是1->2的项目；界面与当时产品设计的界面存在出入；所以更好的做法是直接启动项目看系统里面的页面来进行分析更好。要查看保障管理的界面；需要启动的服务有：

* 前端nginx
* 后端微服务：GatewayMgtStart、InsuranceMgtStart、SecurityOauthStart、SecurityWebStart

访问：http://sf.mgt.itheima.net/#/login ；登录之后；再点击 基础数据——》保障管理





|  |
| --- |
| **保障项分类**：并不是保险产品分类，只有规则和信息两个下拉选项；如果以后遇到产品原型也是类似的，这信息在2个选项以内的话；一般可以写常量或者写死（不需要对这些选项做管理）  **保障项值**：可以设置多个不同的内容（值名称，值，分数）  ① 可以将这些存到一个子表 ----》比较清楚；但是缺点就是以后查询关联比较麻烦  ② 可以设计一个字段存储保障项值，这个值的类型为json ---》比较灵活；尤其是这些内容是在前端展示和使用的，返回给前端的话也方便操作 ----》以后遇到类似的原型和设计，可以采用此方法 |

而在app端点击产品查看详情时，也能够看到这些保障项的使用：

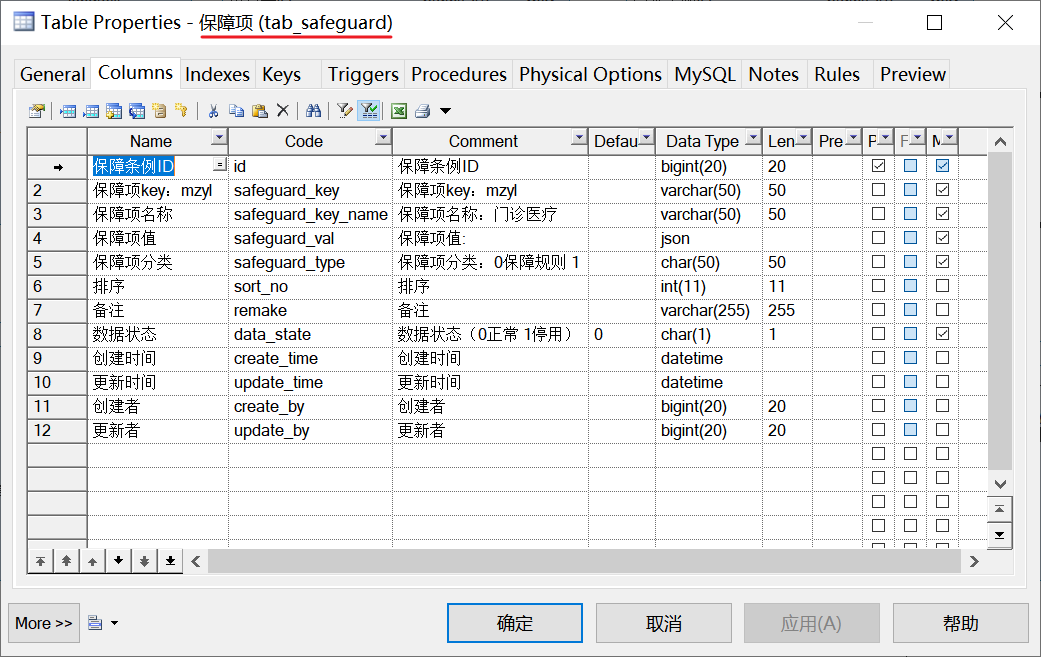


**功能方面**：分页搜索、新建、删除、编辑（前端加载数据，不需要后端实现）、保存编辑、启用/禁用、功能栏的全部/启用/禁用(前端未实现；但也是分页查询)；也就是要实现的功能总共**5**个。

**数据方面**：只是保存编辑页面中的数据、主键、列表中需要的创建时间等。

**2.2、数据库表设计**

按照需求分析中的思路；只需要设计一张表 sfbx-insurance.保障项表tab\_safeguard 具体表结构如下：



|  |
| --- |
| 保障项值；在数据库中存储一个json字符串；json的格式参考如下：  [{"name":“共享400万”,"val":"4000000",“score”:"1"},{"name":“共享300万”,"val":"3000000",“score”:"2"}] |

**2.3、接口设计**

* **方式一**：后端结合原型界面，和自己设计的数据库表；先自己设计，然后再与前端确认。在我们项目中，项目已存在，接口被固定；暂时不太适用。
* **方式二**：在1-2的项目中；接口已经确定，前端也已经使用接口在开发或者前端已经完成；那么作为后端就只需要去追踪出来对应业务模块的接口信息即可；这是更省事的做法。而四方保险属于这种情况。接下来；咱们梳理出来对应分析出来的**5**个接口信息：

**1）分页搜索**

|  |  |
| --- | --- |
| 说明 | 请求信息 |
| **接口地址** | http://sf.mgt.itheima.net/api/insurance-mgt/safeguard/page/{pageNum}/{pageSize} |
| **请求方式** | POST |
| **请求参数** | SafeguardVO 对应的数据结构如：{pageSize: 10, total: 1, pageNum: 1, safeguardKey: "", safeguardKeyName: "身故保险金"} |
| **响应结果** | ResponseResult<Page<SafeguardVO>> 返回的结构参考如下：  {  "code": 200,  "msg": "操作成功",  "data": {  "records": [  {  "id": "1753336375152783362",  "createTime": "2024-02-02 16:35:42",  "updateTime": "2024-05-13 18:44:03",  "createBy": "1371500419615895553",  "updateBy": "1371500419615895553",  "dataState": "0",  "creator": null,  "safeguardKey": "BZ056",  "safeguardKeyName": "身故保险金",  "safeguardVal": "[{\"val\": \"1\", \"name\": \"详见保险条款\", \"score\": \"100\"}]",  "safeguardType": "0",  "sortNo": null,  "remake": null,  "checkedIds": null  }  ],  "total": 1,  "size": 10,  "current": 1,  "orders": [],  "optimizeCountSql": true,  "hitCount": false,  "countId": null,  "maxLimit": null,  "searchCount": true,  "pages": 1  },  "operatorId": 1371500419615895553,  "operatorName": "admin@qq.com",  "operatorSex": "0",  "\_class": "com.baomidou.mybatisplus.extension.plugins.pagination.Page",  "tip": "本站点所有接口仅用于教学演示，所有数据均非真实数据，请勿用作其它用途！",  "operationTime": "2024-05-13 18:44:03"  } |

**2）新建**

|  |  |
| --- | --- |
| 说明 | 请求信息 |
| **接口地址** | http://sf.mgt.itheima.net/api/insurance-mgt/safeguard |
| **请求方式** | PUT ---> 全量更新，如果不存在则新增 |
| **请求参数** | {  "safeguardKeyName": "test111",  "safeguardKey": "111",  "safeguardVal": "[{\"name\":\"111\",\"val\":\"11\",\"score\":\"1\"}]",  "safeguardType": "1",  "icon": "",  "dataState": "0",  "resourceType": "F"  } |
| **响应结果** | ResponseResult<SafeguardVO> 数据结构参考如下：  {  "code": 200,  "msg": "操作成功",  "data": {  "id": "1789999088167677954",  "createTime": "2024-05-13 20:40:13",  "updateTime": "2024-05-13 20:40:13",  "createBy": "1371500419615895553",  "updateBy": "1371500419615895553",  "dataState": "0",  "creator": null,  "safeguardKey": "111",  "safeguardKeyName": "test111",  "safeguardVal": "[{\"name\":\"111\",\"val\":\"11\",\"score\":\"1\"}]",  "safeguardType": "1",  "sortNo": null,  "remake": null,  "checkedIds": null  },  "operatorId": 1371500419615895553,  "operatorName": "admin@qq.com",  "operatorSex": "0",  "\_class": "com.itheima.sfbx.insurance.dto.SafeguardVO",  "tip": "本站点所有接口仅用于教学演示，所有数据均非真实数据，请勿用作其它用途！",  "operationTime": "2024-05-13 20:40:13"  } |

**3）保存编辑**

|  |  |
| --- | --- |
| 说明 | 请求信息 |
| **接口地址** | http://sf.mgt.itheima.net/api/insurance-mgt/safeguard |
| **请求方式** | PATCH ---》部分更新，不存在的话更新失败 |
| **请求参数** | SafeguardVO 数据结构参考如下：  {  "id": "1789999088167677954",  "createTime": "2024-05-13 20:40:14",  "updateTime": "2024-05-13 20:40:14",  "createBy": "1371500419615895553",  "updateBy": "1371500419615895553",  "dataState": "0",  "creator": null,  "safeguardKey": "111",  "safeguardKeyName": "test1112",  "safeguardVal": "[{\"val\":\"11\",\"name\":\"111\",\"score\":\"1\"}]",  "safeguardType": "1",  "sortNo": null,  "remake": null,  "checkedIds": null  } |
| **响应结果** | ResponseResult<Boolean> 数据结构参考如下：  {  "code": 200,  "msg": "操作成功",  "data": true,  "operatorId": 1371500419615895553,  "operatorName": "admin@qq.com",  "operatorSex": "0",  "\_class": "java.lang.Boolean",  "tip": "本站点所有接口仅用于教学演示，所有数据均非真实数据，请勿用作其它用途！",  "operationTime": "2024-05-13 20:45:02"  } |

**4）启用/禁用**

这个功能与 保存编辑是一模一样的接口；所以并不需要再写，调用 保存编辑 的接口即可。

**5）删除**

|  |  |
| --- | --- |
| 说明 | 请求信息 |
| **接口地址** | http://sf.mgt.itheima.net/api/insurance-mgt/safeguard |
| **请求方式** | DELETE |
| **请求参数** | SafeguardVO 只传递了要删除的Id数组；数据结构参考：{checkedIds: ["1789999088167677954"]} |
| **响应结果** | ResponseResult<Boolean>  {  "code": 200,  "msg": "操作成功",  "data": true,  "operatorId": 1371500419615895553,  "operatorName": "admin@qq.com",  "operatorSex": "0",  "\_class": "java.lang.Boolean",  "tip": "本站点所有接口仅用于教学演示，所有数据均非真实数据，请勿用作其它用途！",  "operationTime": "2024-05-13 21:04:36"  } |

**2.4、开发实现**

**2.4.1、分页搜索**

**1）编写处理器**

先删除再创建 insurance-mgt\src\main\java\com\itheima\sfbx\insurance\web\SafeguardController.java内容如下：

|  |
| --- |
| Java package com.itheima.sfbx.insurance.web;  import com.baomidou.mybatisplus.extension.plugins.pagination.Page; import com.github.xiaoymin.knife4j.annotations.ApiOperationSupport; import com.itheima.sfbx.framework.commons.basic.ResponseResult; import com.itheima.sfbx.framework.commons.utils.ResponseResultBuild; import com.itheima.sfbx.insurance.dto.SafeguardVO; import com.itheima.sfbx.insurance.service.ISafeguardService; import io.swagger.annotations.Api; import io.swagger.annotations.ApiImplicitParam; import io.swagger.annotations.ApiImplicitParams; import io.swagger.annotations.ApiOperation; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.web.bind.annotation.\*;  @Api(tags = "保障项") @RestController @RequestMapping("/safeguard") public class SafeguardController {   @Autowired  private ISafeguardService safeguardService;   */\*\**  *\* 分页查询*  *\* @param safeguardVO 查询条件*  *\* @param pageNum 页号*  *\* @param pageSize 页大小*  *\* @return 查询结果*  *\*/* @ApiOperation("分页查询")  */\*swagger 注解说明*  *@ApiImplicitParams 表示能接受的参数列表。*  *@ApiImplicitParam 表示具体的参数说明和属性*  *@ApiOperationSupport 表示swagger的扩展属性，可以对指定参数做扩展补充*  *includeParameters 表示参数safeguardVO中的属性，可以指定哪些属性需要swagger展示*   *\*/*  @ApiImplicitParams({  @ApiImplicitParam(name = "safeguardVO",value = "VO对象",required = true,dataType = "SafeguardVO"),  @ApiImplicitParam(paramType = "path",name = "pageNum",value = "页码",example = "1",dataType = "Integer"),  @ApiImplicitParam(paramType = "path",name = "pageSize",value = "每页条数",example = "10",dataType = "Integer")  })  @ApiOperationSupport(includeParameters = {"safeguardVO.safeguardType","safeguardVO.safeguardKey","safeguardVO.safeguardVal","safeguardVO.remake"})  @PostMapping("/page/{pageNum}/{pageSize}")  public ResponseResult<Page<SafeguardVO>> findSafeguardVOPage(  @RequestBody SafeguardVO safeguardVO,  @PathVariable("pageNum") int pageNum,  @PathVariable("pageSize") int pageSize  ){  Page<SafeguardVO> page = safeguardService.findPage(safeguardVO, pageNum, pageSize);  return ResponseResultBuild.*successBuild*(page);  }   } |

**2）实现业务方法**

因为项目中其它地方有引用到保障项业务接口；所以已经存在 sfbx-insurance\insurance-service\src\main\java\com\itheima\sfbx\insurance\service\ISafeguardService.java 这个接口不需要创建；

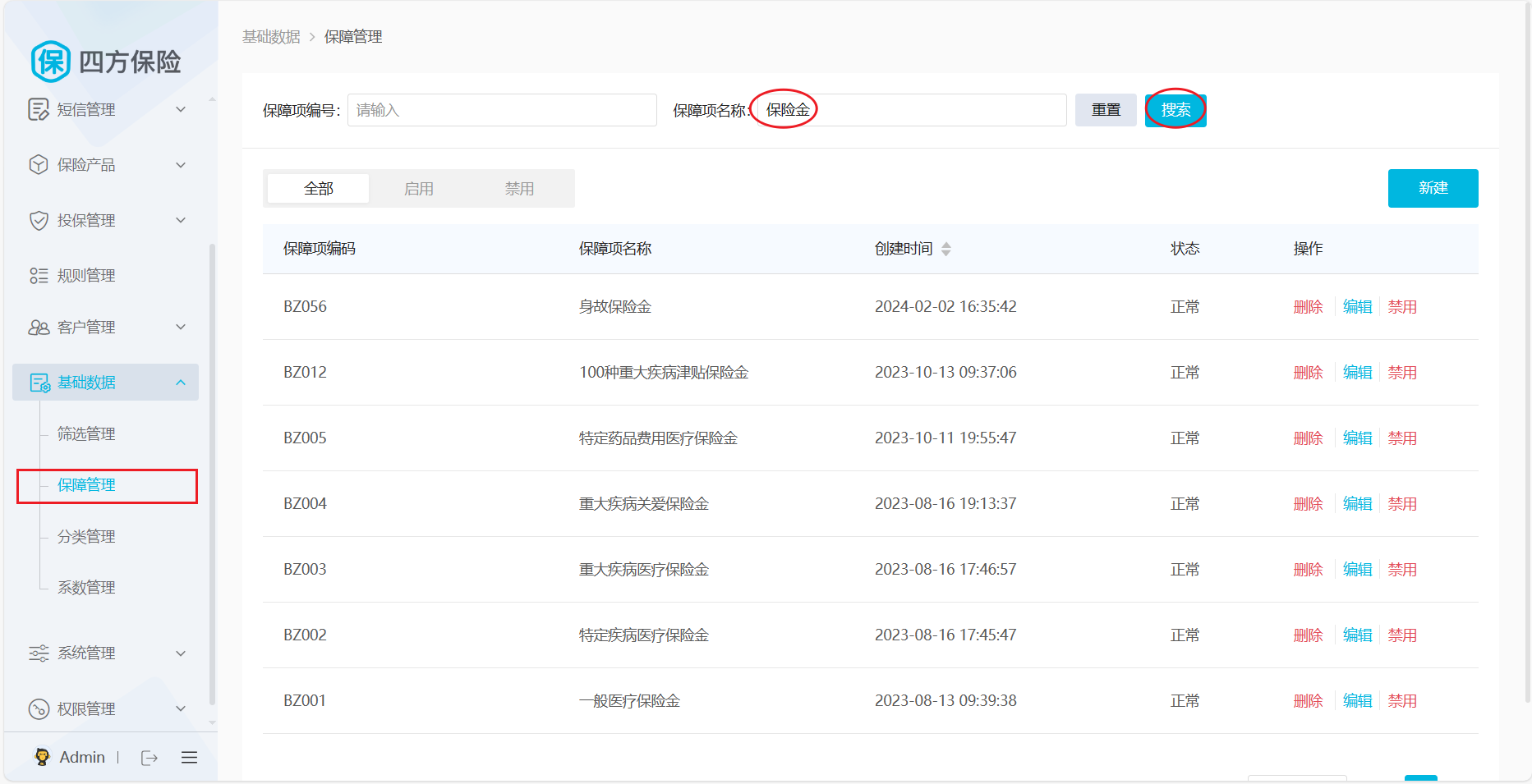
**只需要删除并重编写对应的实现类如下**：

insurance-service\src\main\java\com\itheima\sfbx\insurance\service\impl\SafeguardServiceImpl.java

|  |
| --- |
| Java package com.itheima.sfbx.insurance.service.impl;  import com.baomidou.mybatisplus.core.conditions.query.LambdaQueryWrapper; import com.baomidou.mybatisplus.extension.plugins.pagination.Page; import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl; import com.itheima.sfbx.framework.commons.exception.ProjectException; import com.itheima.sfbx.framework.commons.utils.BeanConv; import com.itheima.sfbx.framework.commons.utils.EmptyUtil; import com.itheima.sfbx.framework.commons.utils.ExceptionsUtil; import com.itheima.sfbx.insurance.dto.SafeguardVO; import com.itheima.sfbx.insurance.enums.SafeguardEnum; import com.itheima.sfbx.insurance.mapper.SafeguardMapper; import com.itheima.sfbx.insurance.pojo.Safeguard; import com.itheima.sfbx.insurance.service.ISafeguardService; import lombok.extern.slf4j.Slf4j; import org.springframework.stereotype.Service;  import java.util.Arrays; import java.util.List; import java.util.stream.Collectors;  @Slf4j @Service public class SafeguardServiceImpl extends ServiceImpl<SafeguardMapper, Safeguard> implements ISafeguardService {  @Override  public Page<SafeguardVO> findPage(SafeguardVO safeguardVO, int pageNum, int pageSize) {  Page<Safeguard> safeguardPage = new Page<>(pageNum, pageSize);  LambdaQueryWrapper<Safeguard> queryWrapper = new LambdaQueryWrapper<>();  *//根据保障项编号* queryWrapper.eq(!EmptyUtil.*isNullOrEmpty*(safeguardVO.getSafeguardKey()),  Safeguard::getSafeguardKey, safeguardVO.getSafeguardKey());  *//根据保障项名称* queryWrapper.like(!EmptyUtil.*isNullOrEmpty*(safeguardVO.getSafeguardKeyName()),  Safeguard::getSafeguardKeyName, safeguardVO.getSafeguardKeyName());  *//状态* queryWrapper.eq(!EmptyUtil.*isNullOrEmpty*(safeguardVO.getDataState()),  Safeguard::getDataState, safeguardVO.getDataState());  *//按照创建时间降序排* queryWrapper.orderByDesc(Safeguard::getCreateTime);   Page<Safeguard> resultPage = page(safeguardPage, queryWrapper);   *//转换为VO* Page<SafeguardVO> safeguardVOPage = BeanConv.*toPage*(resultPage, SafeguardVO.class);  return safeguardVOPage;  }   @Override  public SafeguardVO findById(String safeguardId) {  return null;  }   @Override  public SafeguardVO save(SafeguardVO safeguardVO) {  return null;  }   @Override  public Boolean update(SafeguardVO safeguardVO) {  return null;  }   @Override  public Boolean delete(String[] checkedIds) {  return null;  }   @Override  public List<SafeguardVO> findList(SafeguardVO safeguardVO) {  return null;  }   @Override  public List<SafeguardVO> findShowPageItemByKey(List<String> safeguardKeyList) {  return null;  }   @Override  public SafeguardVO findBySafeguardKey(String safeguardKey) {  return null;  }  } |

**3）测试**

访问 http://sf.mgt.itheima.net/#/base/guarantee 界面中；根据编号、名称能够查询出保障项数据的话则说明开发测试成功。



**2.4.2、新建**

**1）处理器方法**

新增 com.itheima.sfbx.insurance.web.SafeguardController#createSafeguard 方法如下：

|  |
| --- |
| Java */\*\**  *\* 新增保障项*  *\* @param safeguardVO 保障项VO对象*  *\* @return 新增结果*  *\*/* @ApiOperation(value = "新增保障项", notes = "新增保障项") @ApiImplicitParam(name = "safeguardVO",value = "VO对象",required = true,dataType = "SafeguardVO") @ApiOperationSupport(includeParameters = {"safeguardVO.safeguardType","safeguardVO.safeguardKey","safeguardVO.safeguardVal","safeguardVO.remake", "safeguardVO.dataState", "safeguardVO.sortNo"}) @PutMapping public ResponseResult<SafeguardVO> createSafeguard(@RequestBody SafeguardVO safeguardVO){  SafeguardVO result = safeguardService.save(safeguardVO);  return ResponseResultBuild.*successBuild*(result); } |

**2）业务方法**

实现 com.itheima.sfbx.insurance.service.impl.SafeguardServiceImpl#save 方法如下：

|  |
| --- |
| Java @Override public SafeguardVO save(SafeguardVO safeguardVO) {  try {  *//将safeguardVO转换为safeguard* Safeguard safeguard = BeanConv.*toBean*(safeguardVO, Safeguard.class);  *//保存数据* boolean flag = save(safeguard);  if (!flag) {  throw new RuntimeException("保存保障项失败！");  }  *//将safeguard转换为safeguardVO并返回* return BeanConv.*toBean*(safeguard, SafeguardVO.class);   } catch (Exception e) {  *log*.error("保存保障项失败：{}", ExceptionsUtil.*getStackTraceAsString*(e));  throw new ProjectException(SafeguardEnum.*SAVE\_FAIL*);  } } |

**2.4.3、保存编辑**

**1）处理器方法**

新增方法 com.itheima.sfbx.insurance.web.SafeguardController#updateSafeguard 如下：

|  |
| --- |
| Java */\*\**  *\* 修改保障项*  *\* @param safeguardVO 保障项VO对象*  *\* @return 修改结果*  *\*/* @PatchMapping @ApiOperation(value = "修改保障项", notes = "修改保障项") @ApiImplicitParam(name = "safeguardVO",value = "VO对象",required = true,dataType = "SafeguardVO") @ApiOperationSupport(includeParameters = {"safeguardVO.id","safeguardVO.safeguardType","safeguardVO.safeguardKey","safeguardVO.safeguardVal","safeguardVO.remake", "safeguardVO.dataState", "safeguardVO.sortNo"}) public ResponseResult<Boolean> updateSafeguard(@RequestBody SafeguardVO safeguardVO){  Boolean result = safeguardService.update(safeguardVO);  return ResponseResultBuild.*successBuild*(result); } |

|  |
| --- |
| Put 发送完整数据更新数据，如果不存在可根据提交的数据创建  Patch 可发送部分（只包含要更新的字段内容）更新数据 |

**2）业务方法**

新增方法 com.itheima.sfbx.insurance.service.impl.SafeguardServiceImpl#update 如下：

|  |
| --- |
| Java @Override public Boolean update(SafeguardVO safeguardVO) {  try {  *//将safeguardVO转换为safeguard* Safeguard safeguard = BeanConv.*toBean*(safeguardVO, Safeguard.class);   *//更新并返回更新结果* boolean flag = updateById(safeguard);  if (!flag) {  throw new RuntimeException("更新保障项失败！");  }  return flag;  } catch (Exception e) {  *log*.error("更新保障项失败：{}", ExceptionsUtil.*getStackTraceAsString*(e));  throw new ProjectException(SafeguardEnum.*UPDATE\_FAIL*);  } } |

**2.4.4、删除**

**1）处理器方法**

新增方法 com.itheima.sfbx.insurance.web.SafeguardController#deleteSafeguard 内容如下：

|  |
| --- |
| Java */\*\**  *\* 删除保障项*  *\* @param safeguardVO 保障项VO对象*  *\* @return 删除结果*  *\*/* @DeleteMapping @ApiImplicitParam(name = "safeguardVO",value = "VO对象",required = true,dataType = "SafeguardVO") @ApiOperationSupport(includeParameters = {"safeguardVO.checkedIds"}) public ResponseResult<Boolean> deleteSafeguard(@RequestBody SafeguardVO safeguardVO){  Boolean result = safeguardService.delete(safeguardVO.getCheckedIds());  return ResponseResultBuild.*successBuild*(result); } |

**2）业务方法**

新增方法 com.itheima.sfbx.insurance.web.SafeguardController#deleteSafeguard 内容如下：

|  |
| --- |
| Java @Override public Boolean delete(String[] checkedIds) {  try {  List<Long> idsList = Arrays.*asList*(checkedIds).stream().map(Long::new).collect(Collectors.*toList*());  boolean flag = removeByIds(idsList);  if (!flag) {  throw new RuntimeException("批量删除保障项失败！");  }  return flag;  } catch (RuntimeException e) {  *log*.error("删除保障项异常；{}", ExceptionsUtil.*getStackTraceAsString*(e));  throw new ProjectException(SafeguardEnum.*DEL\_FAIL*);  } } |

**2.5、小结**

**保障项的作用：**

* 用户在购买保险时，保险产品所包含的具体风险或事件
* 风险或事件发生时，保险公司愿意提供的经济赔偿或保障

**保障项的数据结构-示例：**

* 保障项名称：住院医疗费用
* 保障项key： B001-ZY
* 保障项值： [{"name":“共享400万”,"val":"4000000",“score”:"1"},{"name":“共享300万”,"val":"3000000",“score”:"2"}]

**3、系数项-保险保费定价**

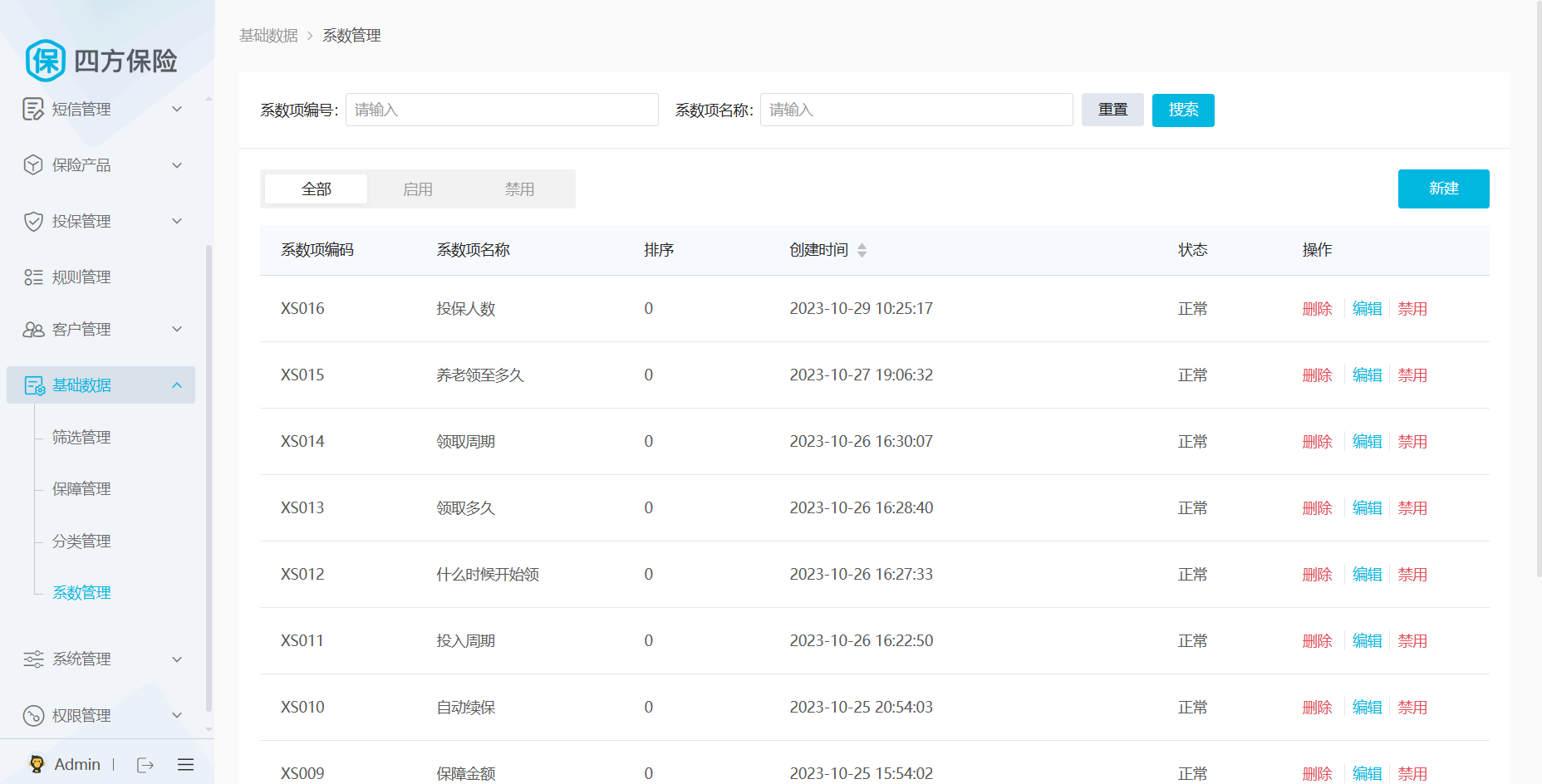
课堂练习该模块

**3.1、需求分析**

要查看系数管理的界面；需要启动的服务有：

* 前端nginx
* 后端微服务：GatewayMgtStart、InsuranceMgtStart、SecurityOauthStart、SecurityWebStart

访问：http://sf.mgt.itheima.net/#/login ；登录之后；再点击 基础数据——》系数管理





而在app端点击产品查看详情时，也能够看到这些系数项的使用：

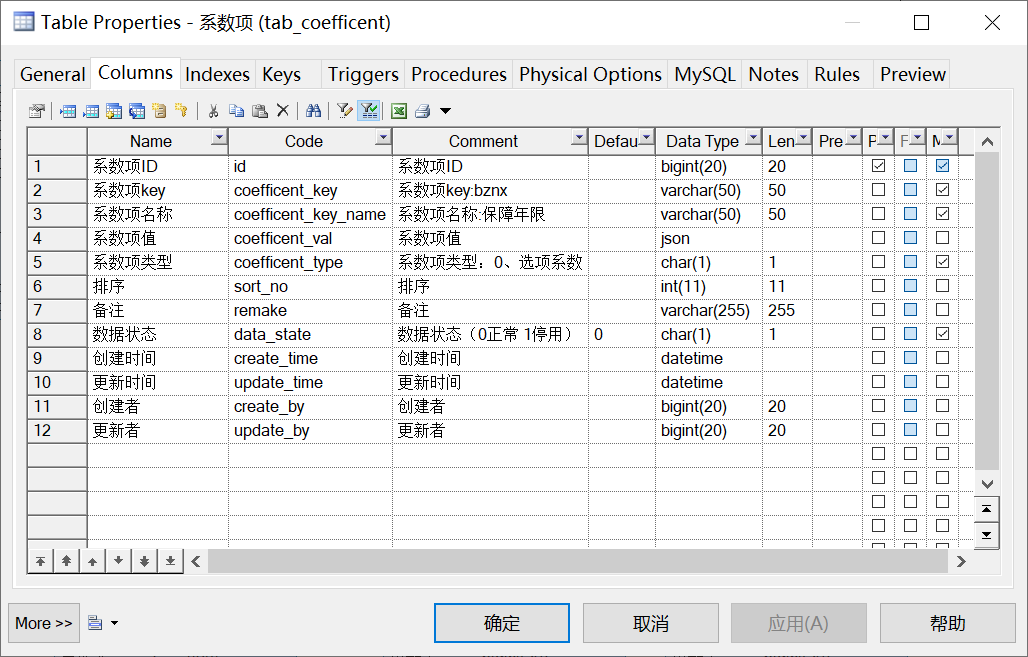


**功能方面**：分页搜索、新建、删除、编辑（前端加载数据，不需要后端实现）、保存编辑、启用/禁用、功能栏的全部/启用/禁用(前端未实现；但也是分页查询)；也就是要实现的功能总共**5**个。

**数据方面**：只是保存编辑页面中的数据、主键、列表中需要的创建时间等。

**3.2、数据库表设计**

按照需求分析中的思路；只需要设计一张表 系数表tab\_coefficent 具体表结构如下：



|  |
| --- |
| 系数项值；在数据库中存储一个json字符串；json的格式参考如下：  [{"name":“10年”,"val":"10",“calculatedVal”:"10"},{"name":“20年”,"val":"20",“calculatedVal”:"20"}] |

**3.3、接口设计**

**1）分页搜索**

|  |  |
| --- | --- |
| 说明 | 请求信息 |
| 接口地址 | http://sf.mgt.itheima.net/api/insurance-mgt/coefficent/page/{pageNum}/{pageSize} |
| **请求方式** | POST |
| **请求参数** | CoefficentVO 对应的数据结构如：  {pageSize: 10, total: 15, pageNum: 1, coefficentKey: "", coefficentKeyName: "投保人数", insuranceType: ""} |
| **响应结果** | ResponseResult<Page<CoefficentVO>> 返回的结构参考如下：  {  "code": 200,  "msg": "操作成功",  "data": {  "records": [  {  "id": "1718453920591900673",  "createTime": "2023-10-29 10:25:17",  "updateTime": "2023-11-30 16:40:55",  "createBy": "1371500419615895553",  "updateBy": "1371500419615895553",  "dataState": "0",  "creator": null,  "coefficentKey": "XS016",  "coefficentKeyName": "投保人数",  "coefficentVal": "[{\"end\": \"\", \"start\": \"\", \"endUnit\": \"\", \"startUnit\": \"\"}]",  "sortNo": 0,  "remake": null,  "coefficentType": "1",  "checkedIds": null  }  ],  "total": 1,  "size": 10,  "current": 1,  "orders": [],  "optimizeCountSql": true,  "hitCount": false,  "countId": null,  "maxLimit": null,  "searchCount": true,  "pages": 1  },  "operatorId": 1371500419615895553,  "operatorName": "admin@qq.com",  "operatorSex": "0",  "\_class": "com.baomidou.mybatisplus.extension.plugins.pagination.Page",  "tip": "本站点所有接口仅用于教学演示，所有数据均非真实数据，请勿用作其它用途！",  "operationTime": "2024-05-15 18:11:12"  } |

**2）新建**

|  |  |
| --- | --- |
| 说明 | 请求信息 |
| **接口地址** | http://sf.mgt.itheima.net/api/insurance-mgt/coefficent |
| **请求方式** | PUT |
| **请求参数** | {  "coefficentKeyName": "test系数名称",  "coefficentKey": "123",  "dataState": "0",  "coefficentType": "1",  "sortNo": 0,  "coefficentVal": "[{\"start\":\"\",\"startUnit\":\"\",\"end\":\"\",\"endUnit\":\"\"}]"  } |
| **响应结果** | ResponseResult<CoefficentVO> 数据结构参考如下：  {  "code": 200,  "msg": "操作成功",  "data": {  "id": "1790686821181591553",  "createTime": "2024-05-15 18:13:02",  "updateTime": "2024-05-15 18:13:02",  "createBy": "1371500419615895553",  "updateBy": "1371500419615895553",  "dataState": "0",  "creator": null,  "coefficentKey": "123",  "coefficentKeyName": "test系数名称",  "coefficentVal": "[{\"start\":\"\",\"startUnit\":\"\",\"end\":\"\",\"endUnit\":\"\"}]",  "sortNo": 0,  "remake": null,  "coefficentType": "1",  "checkedIds": null  },  "operatorId": 1371500419615895553,  "operatorName": "admin@qq.com",  "operatorSex": "0",  "\_class": "com.itheima.sfbx.insurance.dto.CoefficentVO",  "tip": "本站点所有接口仅用于教学演示，所有数据均非真实数据，请勿用作其它用途！",  "operationTime": "2024-05-15 18:13:02"  } |

**3）保存编辑**

|  |  |
| --- | --- |
| 说明 | 请求信息 |
| **接口地址** | http://sf.mgt.itheima.net/api/insurance-mgt/cofficent |
| **请求方式** | PATCH |
| **请求参数** | CoefficentVO数据结构参考如下：  {  "id": "1790686821181591553",  "createTime": "2024-05-15 18:13:02",  "updateTime": "2024-05-15 18:13:02",  "createBy": "1371500419615895553",  "updateBy": "1371500419615895553",  "dataState": "0",  "creator": null,  "coefficentKey": "123",  "coefficentKeyName": "test系数名称22",  "coefficentVal": "[{\"end\":\"\",\"start\":\"\",\"endUnit\":\"\",\"startUnit\":\"\"}]",  "sortNo": 0,  "remake": null,  "coefficentType": "1",  "checkedIds": null  } |
| **响应结果** | ResponseResult<Boolean> 数据结构参考如下：  {  "code": 200,  "msg": "操作成功",  "data": true,  "operatorId": 13715004196158955xx,  "operatorName": "admin@qq.com",  "operatorSex": "0",  "\_class": "java.lang.Boolean",  "tip": "本站点所有接口仅用于教学演示，所有数据均非真实数据，请勿用作其它用途！",  "operationTime": "2024-05-13 20:45:12"  } |

**4）启用/禁用**

这个功能与 保存编辑是一模一样的接口；所以并不需要再写，调用 保存编辑 的接口即可。

**5）删除**

|  |  |
| --- | --- |
| 说明 | 请求信息 |
| **接口地址** | http://sf.mgt.itheima.net/api/insurance-mgt/coefficent |
| **请求方式** | DELETE |
| **请求参数** | CoefficentVO 只传递了要删除的Id数组；数据结构参考：{checkedIds: ["1790686821181591553"]} |
| **响应结果** | ResponseResult<Boolean>  {  "code": 200,  "msg": "操作成功",  "data": true,  "operatorId": 13715004196158955yy,  "operatorName": "admin@qq.com",  "operatorSex": "0",  "\_class": "java.lang.Boolean",  "tip": "本站点所有接口仅用于教学演示，所有数据均非真实数据，请勿用作其它用途！",  "operationTime": "2024-05-13 21:04:46"  } |

**3.4、开发实现**

由于系数项的处理器已经在工程中存在；也就是这个类：com.itheima.sfbx.insurance.web.CoefficentController 里面已经包含了对应接口的处理方法；所以接下来的实现可以删除之后重写处理器方法；对应的业务对象亦如此。

**1）分页搜索**

在 com.itheima.sfbx.insurance.service.impl.CoefficentServiceImpl 里面实现方法如下：

|  |
| --- |
| Java */\*\*\** *\* 系数项多条件组合* *\* @param coefficentVO 系数项* *\* @return QueryWrapper查询条件* *\*/* private QueryWrapper<Coefficent> queryWrapper(CoefficentVO coefficentVO){  QueryWrapper<Coefficent> queryWrapper = new QueryWrapper<>();  *//系数键名称查询* queryWrapper.lambda().eq(!EmptyUtil.*isNullOrEmpty*(coefficentVO.getCoefficentKeyName()),  Coefficent::getCoefficentKeyName, coefficentVO.getCoefficentKeyName());  *//系数键查询* queryWrapper.lambda().eq(!EmptyUtil.*isNullOrEmpty*(coefficentVO.getCoefficentKey()),  Coefficent::getCoefficentKey, coefficentVO.getCoefficentKey());  *//按创建时间降序* queryWrapper.lambda().orderByDesc(Coefficent::getCreateTime);  return queryWrapper; }  @Override @Cacheable(value = CoefficentCacheConstant.*PAGE*,key ="#pageNum+'-'+#pageSize+'-'+#coefficentVO.hashCode()") public Page<CoefficentVO> findPage(CoefficentVO coefficentVO, int pageNum, int pageSize) {  try {  *//构建分页对象* Page<Coefficent> coefficentPage = new Page<>(pageNum, pageSize);   *//构建查询条件* QueryWrapper<Coefficent> queryWrapper = queryWrapper(coefficentVO);   *//执行分页查询* Page<Coefficent> page = page(coefficentPage, queryWrapper);   *//返回结果* return BeanConv.*toPage*(page,CoefficentVO.class);  }catch (Exception e){  *log*.error("系数项分页查询异常：{}", ExceptionsUtil.*getStackTraceAsString*(e));  throw new ProjectException(CoefficentEnum.*PAGE\_FAIL*);  } } |

**2）新建**

在 com.itheima.sfbx.insurance.service.impl.CoefficentServiceImpl 里面实现方法如下：

|  |
| --- |
| Java @Override @Transactional @Caching(evict = {@CacheEvict(value = CoefficentCacheConstant.*PAGE*,allEntries = true),  @CacheEvict(value = CoefficentCacheConstant.*LIST*,allEntries = true)},  put={@CachePut(value =CoefficentCacheConstant.*BASIC*,key = "#result.id")}) public CoefficentVO save(CoefficentVO coefficentVO) {  try {  *//转换CoefficentVO为Coefficent* Coefficent coefficent = BeanConv.*toBean*(coefficentVO,Coefficent.class);   *//保存* boolean flag = save(coefficent);  if (!flag) {  throw new RuntimeException("保存系数失败！");  }   *//转换返回对象CoefficentVO* return BeanConv.*toBean*(coefficent,CoefficentVO.class);  }catch (Exception e){  *log*.error("保存系数项异常：{}", ExceptionsUtil.*getStackTraceAsString*(e));  throw new ProjectException(CoefficentEnum.*SAVE\_FAIL*);  } } |

**3）保存编辑**

在 com.itheima.sfbx.insurance.service.impl.CoefficentServiceImpl 里面实现方法如下：

|  |
| --- |
| Java @Override @Transactional @Caching(evict = {@CacheEvict(value = CoefficentCacheConstant.*PAGE*,allEntries = true),  @CacheEvict(value = CoefficentCacheConstant.*LIST*,allEntries = true),  @CacheEvict(value = CoefficentCacheConstant.*BASIC*,key = "#coefficentVO.id")}) public Boolean update(CoefficentVO coefficentVO) {  try {  *//转换CoefficentVO为Coefficent* Coefficent coefficent = BeanConv.*toBean*(coefficentVO,Coefficent.class);   *//保存* boolean flag = updateById(coefficent);  if (!flag) {  throw new RuntimeException("保存系数失败！");  }   return true;  }catch (Exception e){  *log*.error("修改系数项异常：{}", ExceptionsUtil.*getStackTraceAsString*(e));  throw new ProjectException(CoefficentEnum.*UPDATE\_FAIL*);  } } |

**4）启用/禁用**

与编辑功能是一样的；共用了编辑的实现方法，所以不用再实现。

**5）删除**

在 com.itheima.sfbx.insurance.service.impl.CoefficentServiceImpl 里面实现方法如下：

|  |
| --- |
| Java  @Override @Transactional @Caching(evict = {@CacheEvict(value = CoefficentCacheConstant.*PAGE*,allEntries = true),  @CacheEvict(value = CoefficentCacheConstant.*LIST*,allEntries = true),  @CacheEvict(value = CoefficentCacheConstant.*BASIC*,allEntries = true)}) public Boolean delete(String[] checkedIds) {  try {  List<Long> idsList = Arrays.*stream*(checkedIds).map(Long::*parseLong*).collect(Collectors.*toList*());  boolean flag = removeByIds(idsList);  if (!flag) {  throw new RuntimeException("删除系数失败！");  }  return true;  }catch (Exception e){  *log*.error("删除系数项异常：{}", ExceptionsUtil.*getStackTraceAsString*(e));  throw new ProjectException(CoefficentEnum.*DEL\_FAIL*);  } } |

**3.5、小结**

**系数项的作用：**

* 用户在购买保险时，通过系数项的选择会有不同的保费报价，系数是影响投保价格的唯一因素

**系数项的数据结构-示例：**

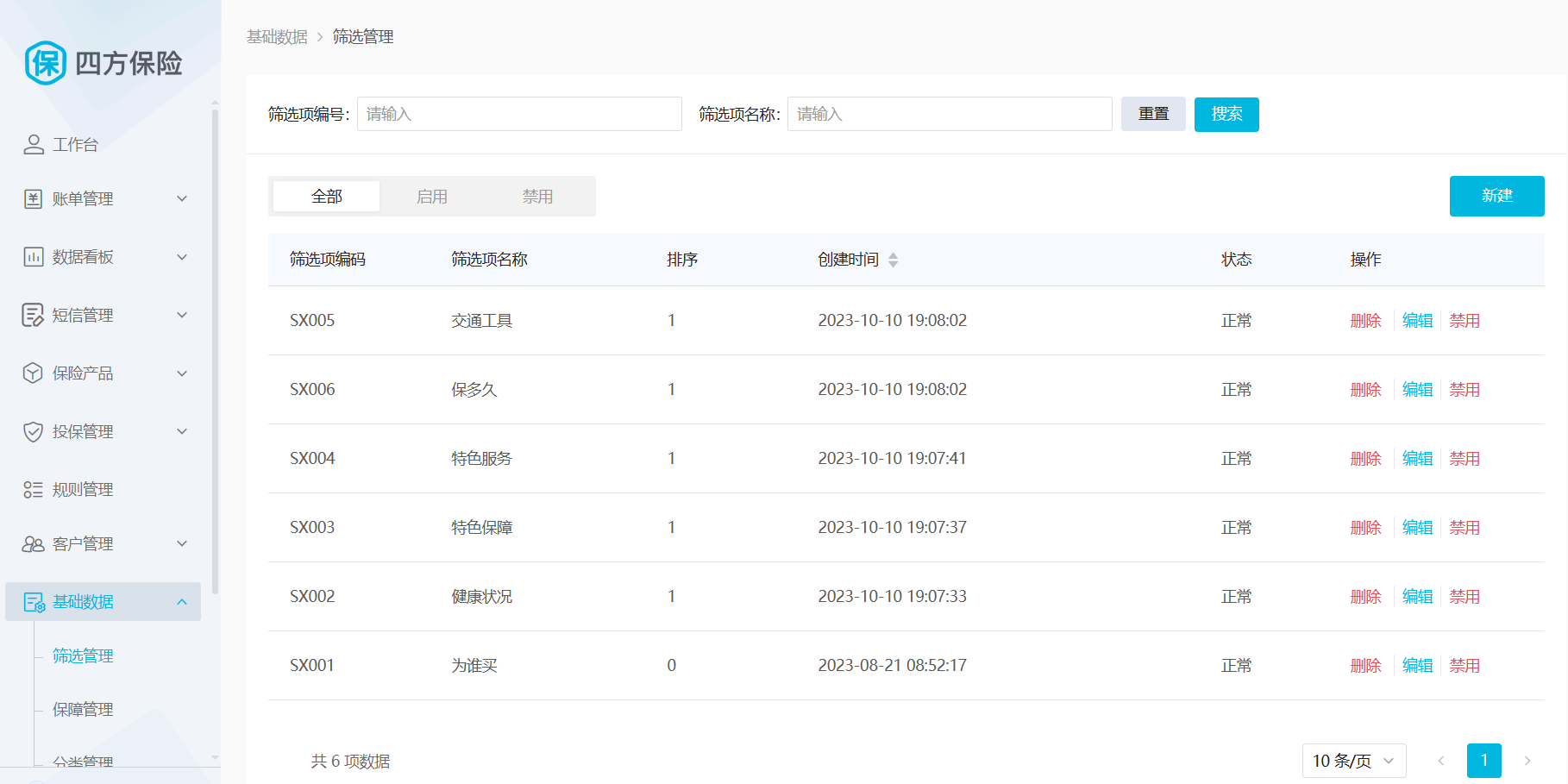
* 系数项名称：保障年限
* 系数项key： X001-BZNX
* 系数项值格式：
* 选项系数：[{"name":“10年”,"val":"10",“calculatedVal”:"10"},{"name":“20年”,"val":"20",“calculatedVal”:"20"}]
* 范围系数：[{"end": "10", "start": "28", "endUnit": "YEAR", "startUnit": "DAY"}, {"end": "20", "val": "", "name": "", "start": "10", "endUnit": "YEAR", "startUnit": "YEAR", "calculatedVal": ""}, {"end": "30", "val": "", "name": "", "start": "20", "endUnit": "YEAR", "startUnit": "YEAR", "calculatedVal": ""}]
* 年限系数：[ {"val": "20YEAR", "name": "20年", "unit": "YEAR", "calculatedVal": "20"},{"val": "30YEAR", "name": "30年", "unit": "YEAR", "calculatedVal": "30"}]

**4、筛选项-查找保险产品**

与前面的两个是一样的做法；阅读该模块的实现即可。

**4.1、需求分析**

登录后台管理系统——》基础数据——》筛选管理



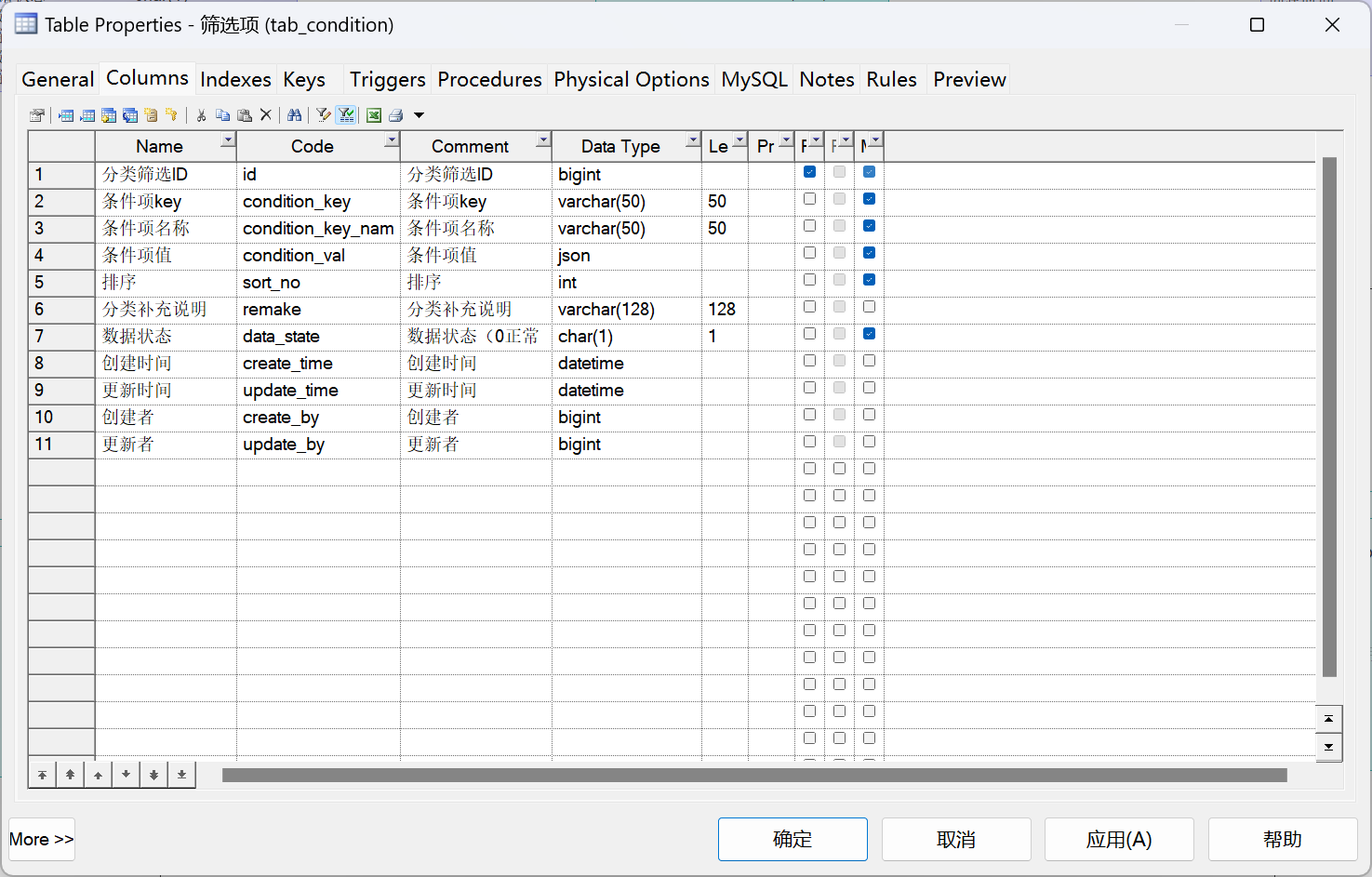


而在app端点击产品查看详情时，也能够看到这些筛选项的使用：



**4.2、数据库表设计**

按照需求分析中的思路；只需要设计一张表 筛选项表tab\_condition 具体表结构如下：



|  |
| --- |
| 筛选项值；在数据库中存储一个json字符串；json的格式参考如下：  [{"name":“有其他慢性病”,"val":"MXB"},{"name":“健康”,"val":"JK"}] |

**4.3、接口设计**

项目中已经引入了 knife4j（swagger）;所以可以直接访问接口管理平台：http://sf.mgt.itheima.net:7066/doc.html#/home 这样可以更加直观的查看、调试接口。



**4.4、测试**

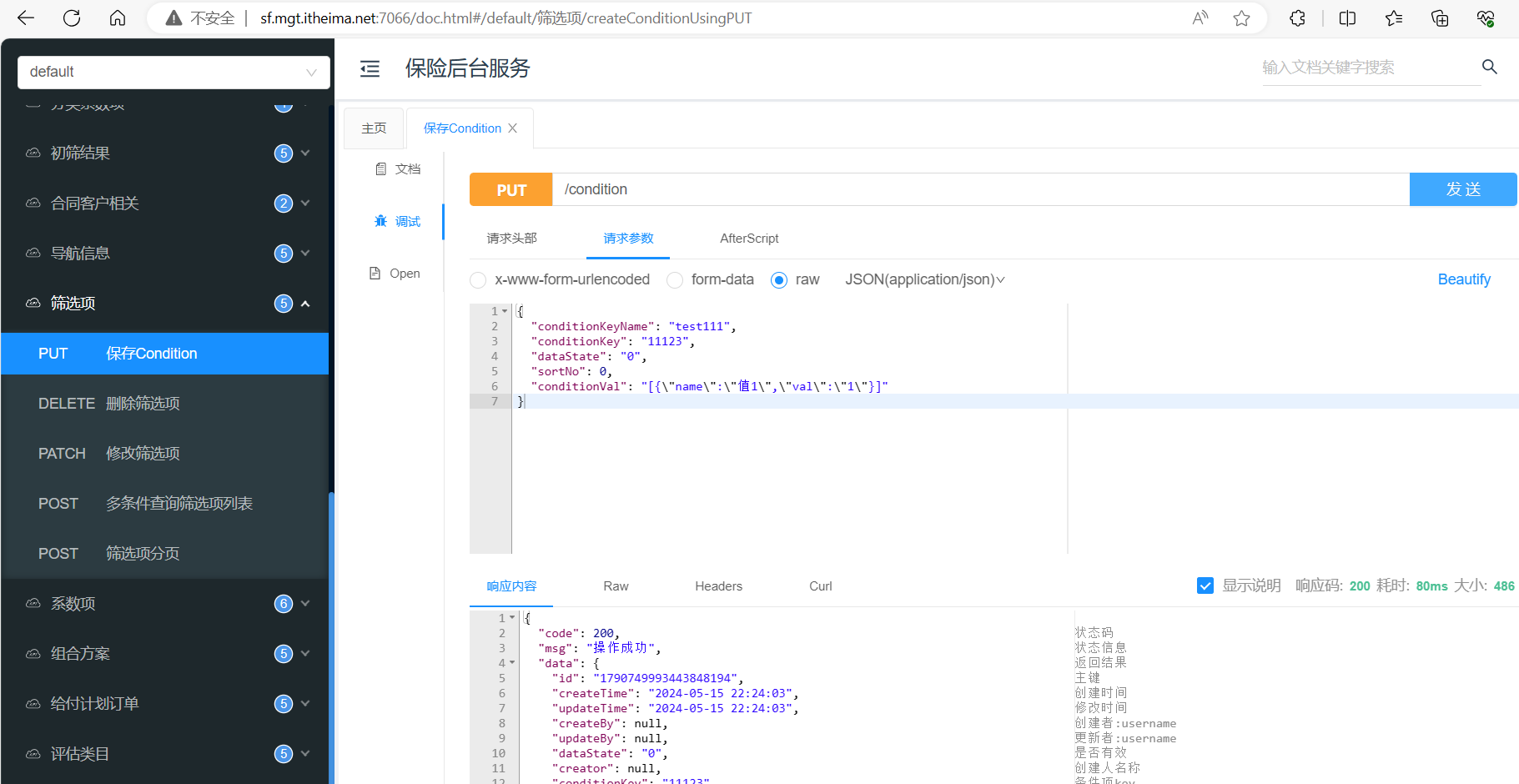
既然我们在这个模块中已实现全部功能，那么只需要在测试管理平台中进行功能的确认测试。

**1）新建**

测试地址：http://sf.mgt.itheima.net:7066/doc.html#/default/%E7%AD%9B%E9%80%89%E9%A1%B9/createConditionUsingPUT

测试数据参考如下：

|  |
| --- |
| JSON {  "conditionKeyName": "test111",  "conditionKey": "11123",  "dataState": "0",  "sortNo": 0,  "conditionVal": "[{\"name\":\"值1\",\"val\":\"1\"}]" } |



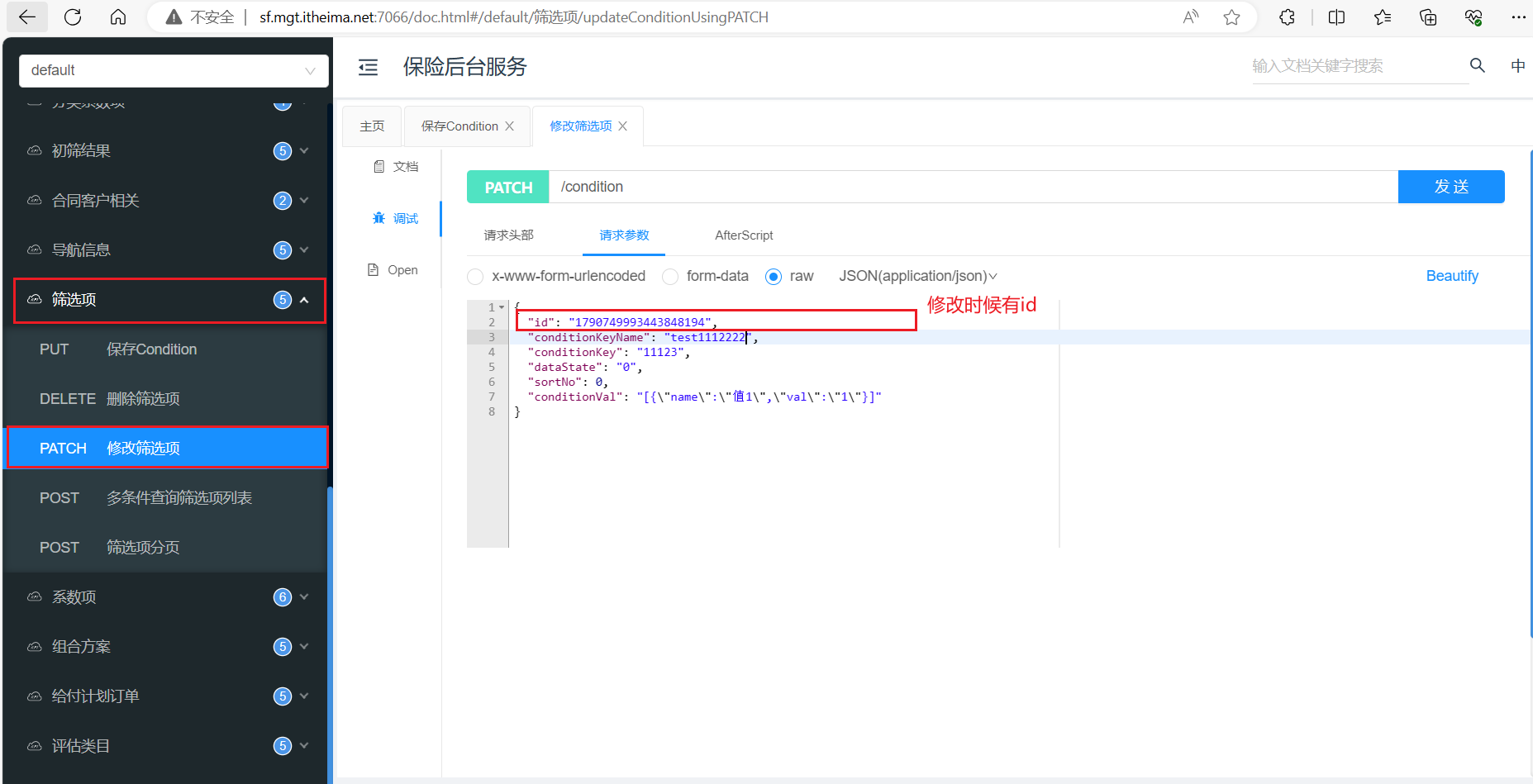
**2）编辑**

测试地址：

http://sf.mgt.itheima.net:7066/doc.html#/default/%E7%AD%9B%E9%80%89%E9%A1%B9/updateConditionUsingPATCH

测试数据参考：

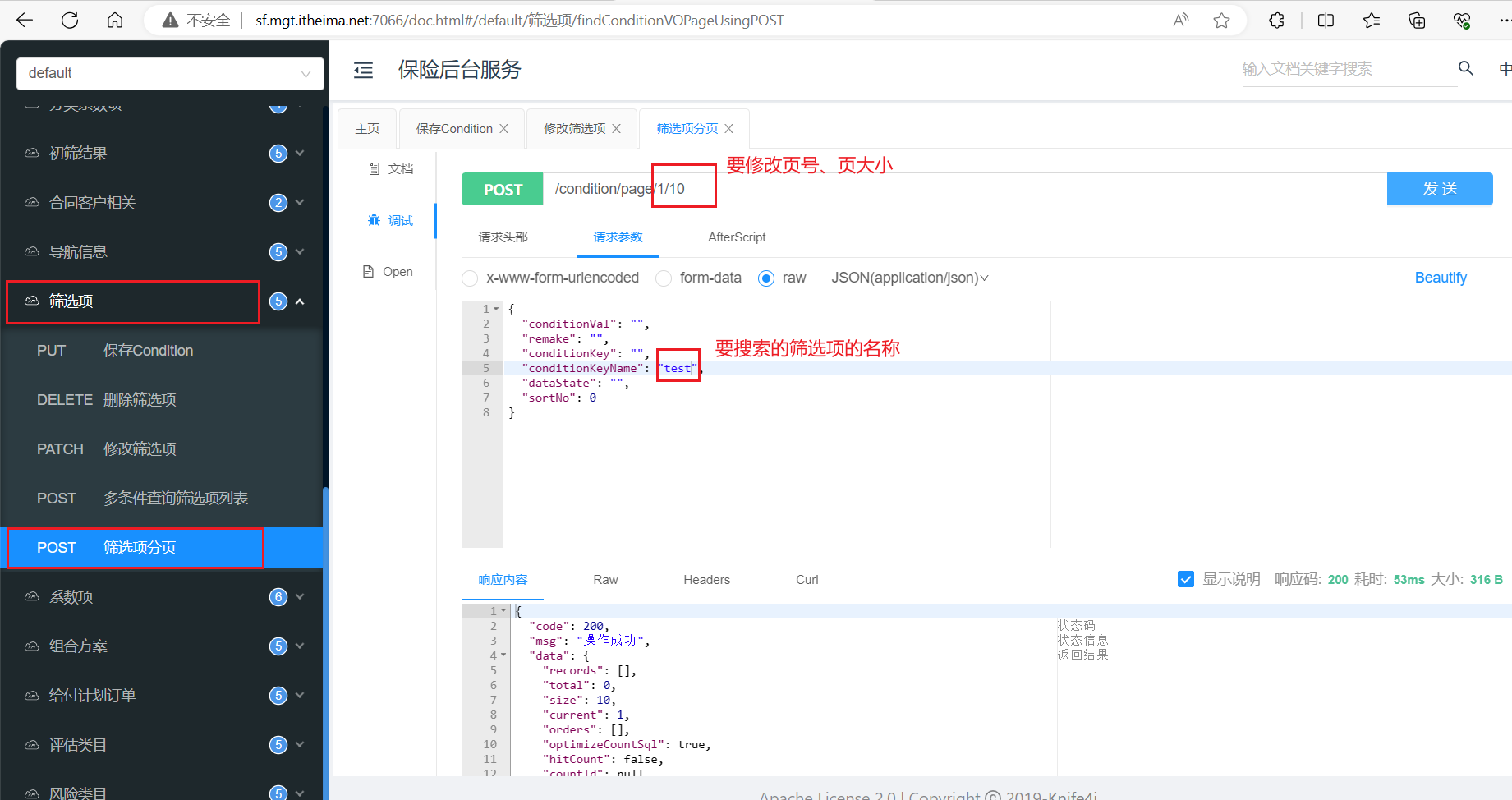
|  |
| --- |
| JSON {  "id": "1790749993443848194要替换为刚刚创建的数据对应的id",  "conditionKeyName": "test1112222",  "conditionKey": "11123",  "dataState": "0",  "sortNo": 0,  "conditionVal": "[{\"name\":\"值1\",\"val\":\"1\"}]" } |



**3）搜索**

测试地址：

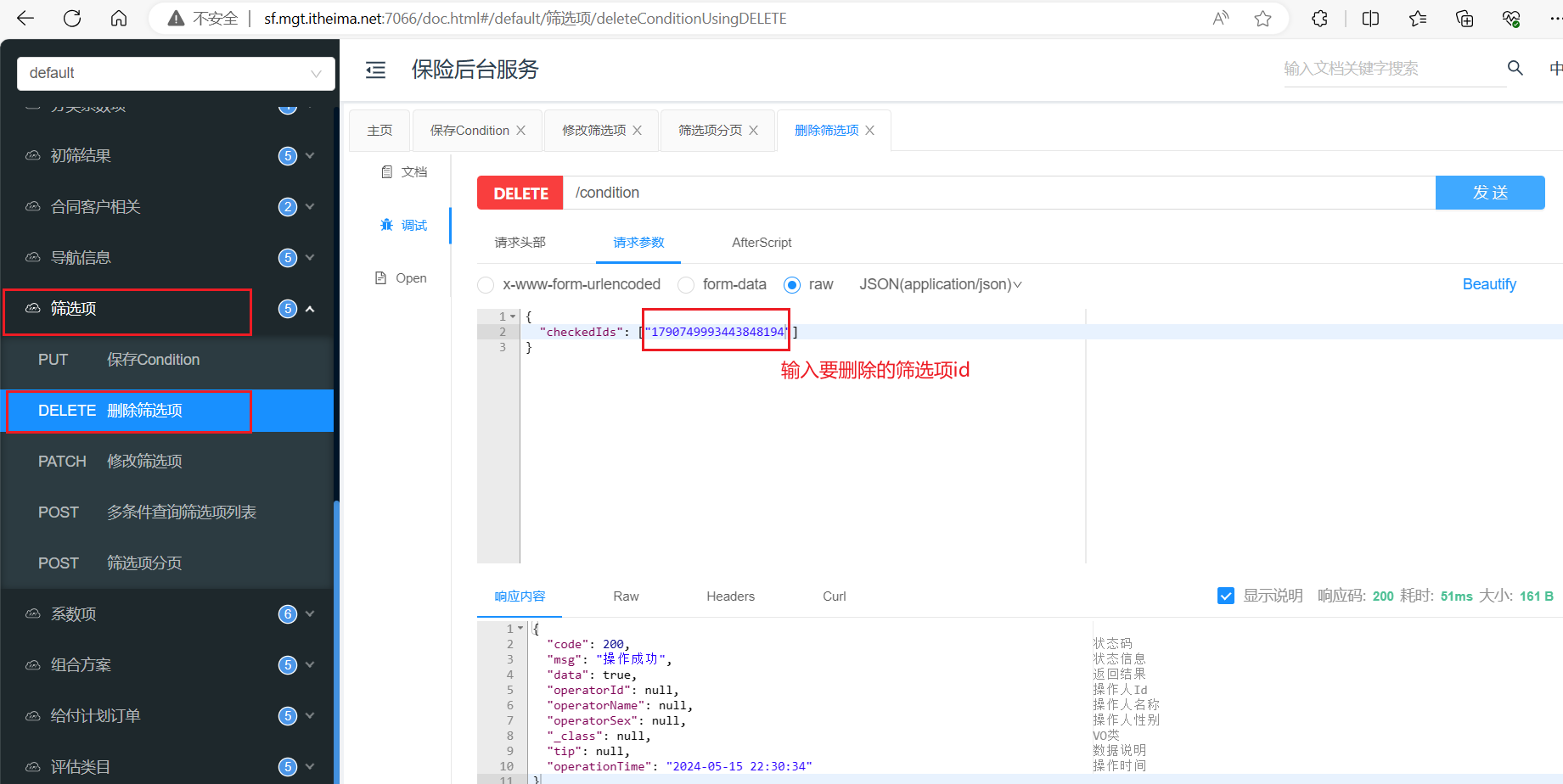
http://sf.mgt.itheima.net:7066/doc.html#/default/%E7%AD%9B%E9%80%89%E9%A1%B9/findConditionVOPageUsingPOST



**4）删除**

测试地址：

http://sf.mgt.itheima.net:7066/doc.html#/default/%E7%AD%9B%E9%80%89%E9%A1%B9/deleteConditionUsingDELETE



**4.5、小结**

**筛选项的作用：**

* 用户在购买保险时，通过筛选项的选择满足自己需求的保险。

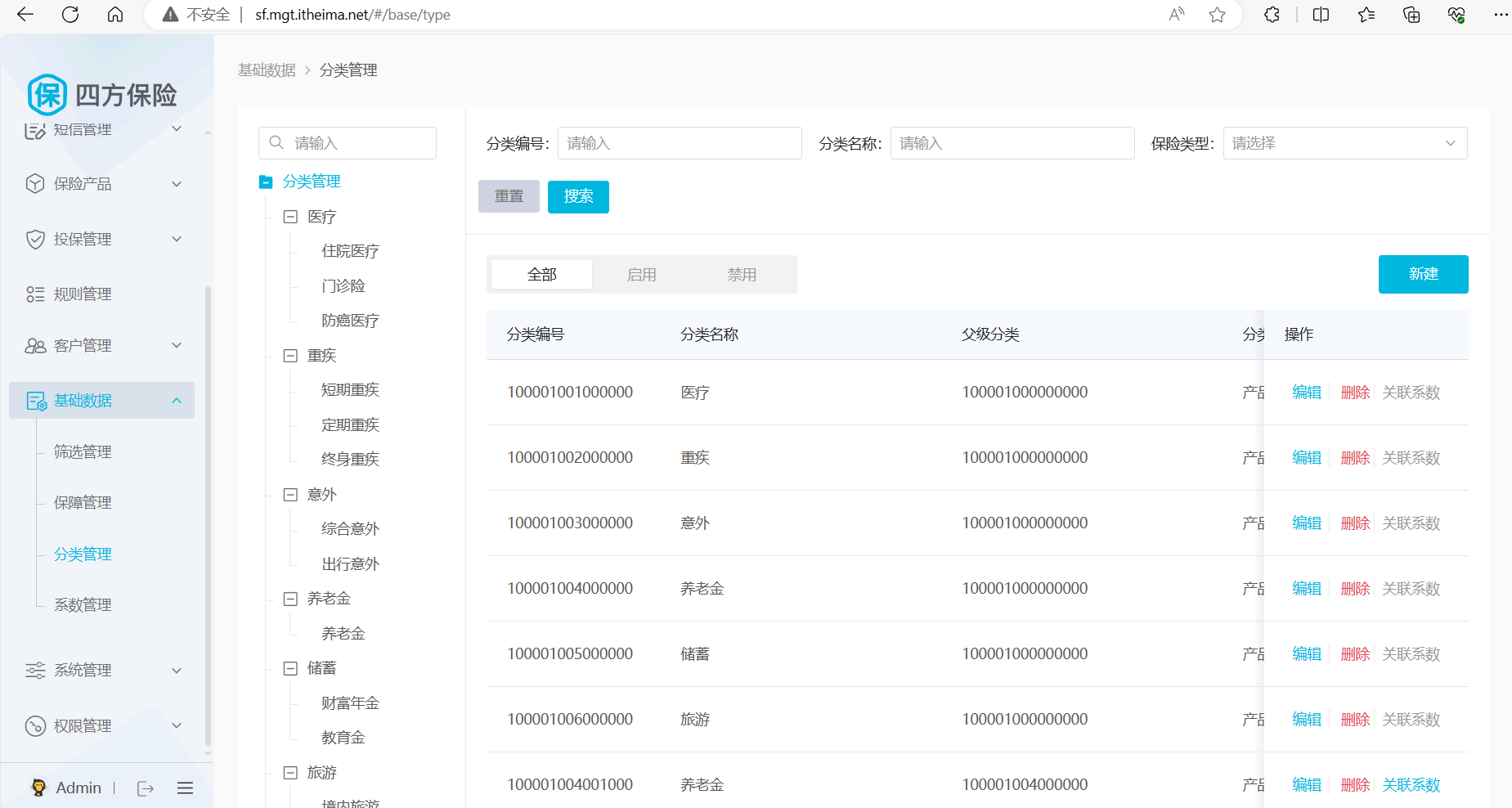
**筛选项的数据结构：**

* 筛选项名称：健康情况
* 筛选项key： S001-JKQK
* 筛选项值： [{"name":“有其他慢性病”,"val":"MXB"},{"name":“健康”,"val":"JK"}]

**5、保险分类-保险产品归属**

**5.1、需求分析**

在后台管理端中分类管理：



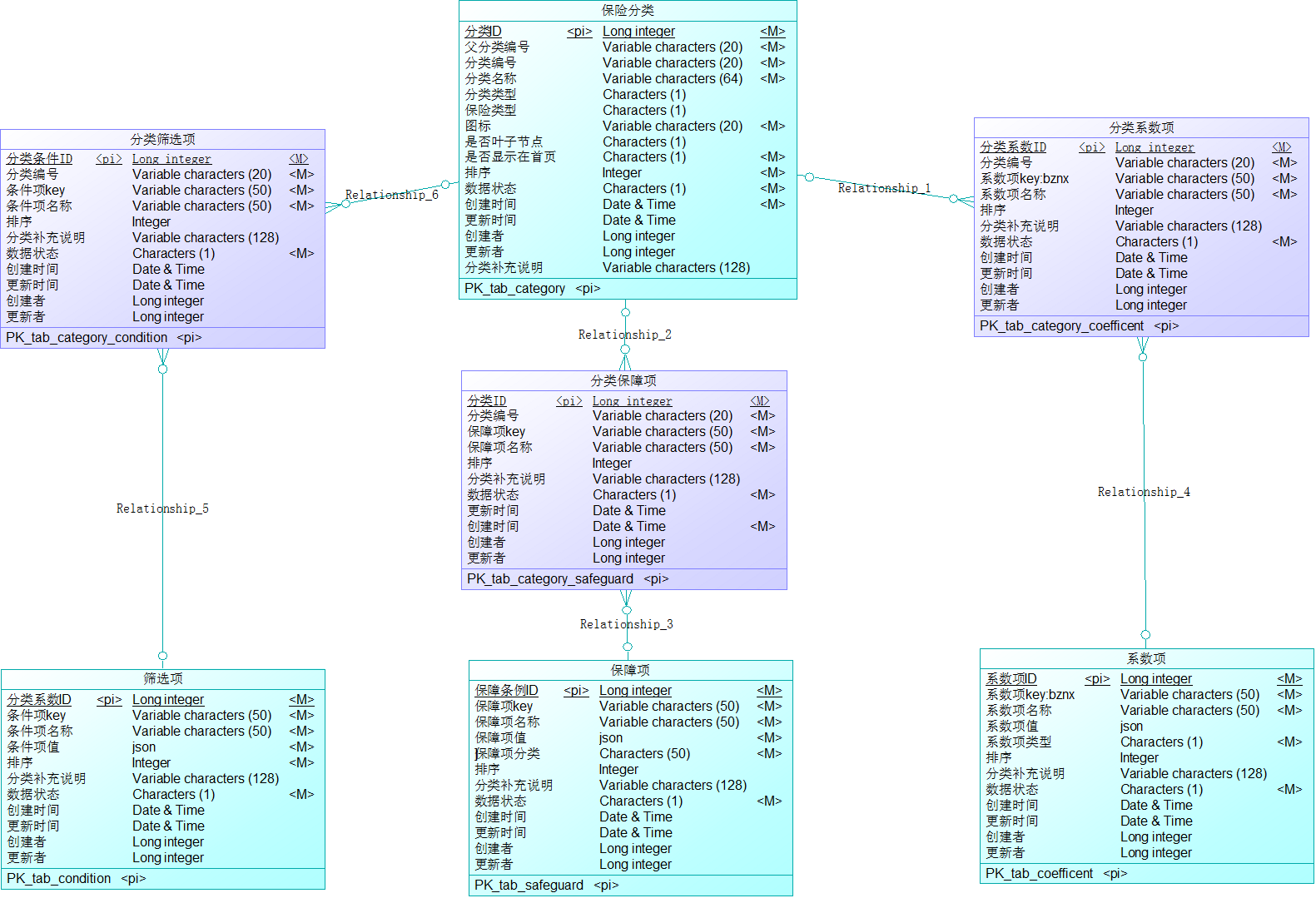
|  |
| --- |
| 上述的 关联系数 只能关联到叶子节点上；比如：储蓄 是 3级分类，它还有 财富年金、教育金两个叶子节点；它自己是不能关联系数等的。 但是 财富年金 是叶子节点则可以关联系数 |

在app端中分类使用的场景：



**5.2、数据库表设计**

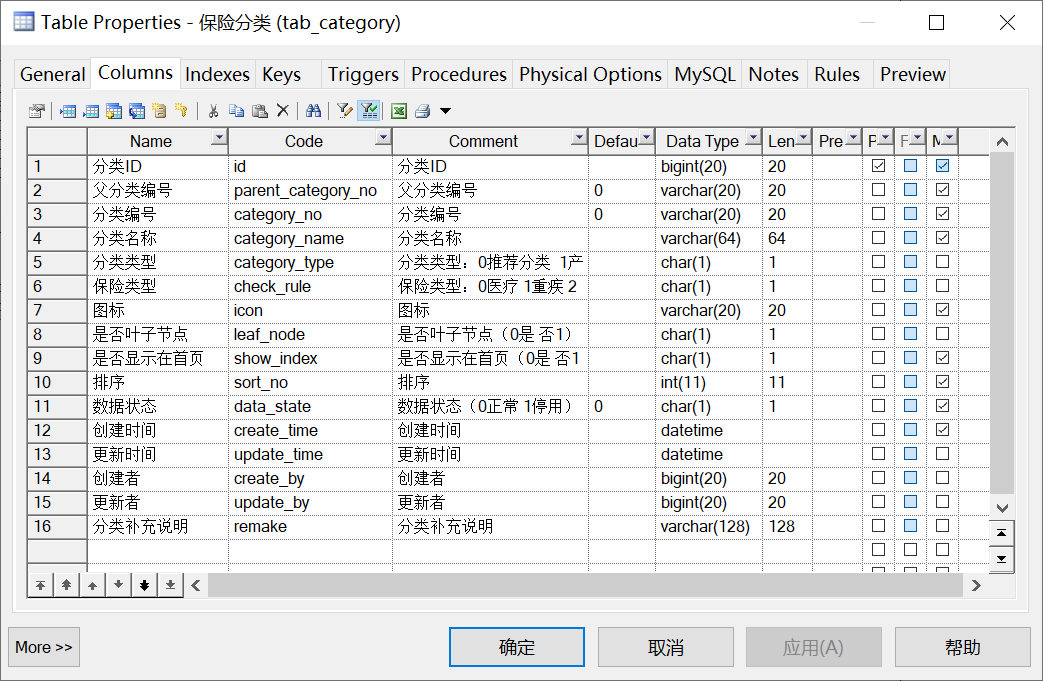
分类与关联关系的总体关系为：



**5.2.1、分类**

查看界面发现；分类有层级关系，且最多4级——》设计一张存储层级关系的数据库表 tab\_category（红色的部分）





**5.2.2、关联关系**

* **分类与保障**

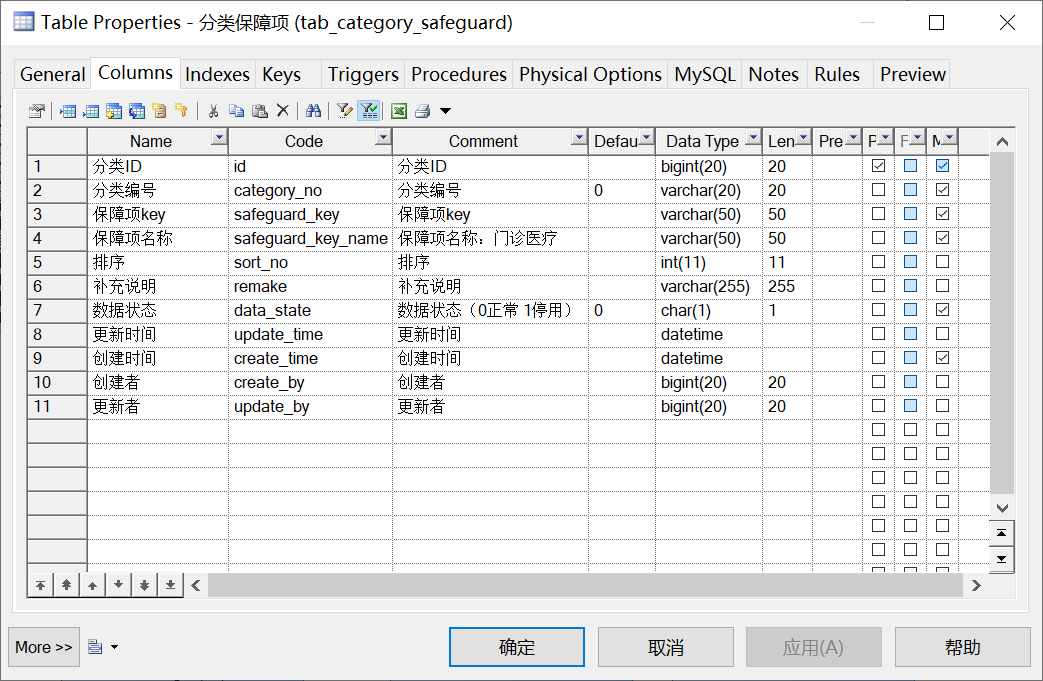
一个分类可以对应管理多个不同的保障项 ===》一对多

一个保障项可以被不同的分类关联 ===》一对多

双向一对多；**说明分类与保障项的关系为：多对多** ===》需要设计中间表tab\_category\_safeguard来保存 分类与保障的关系



与分类的关系如上图；具体的表结构如下：



* **分类与系数**

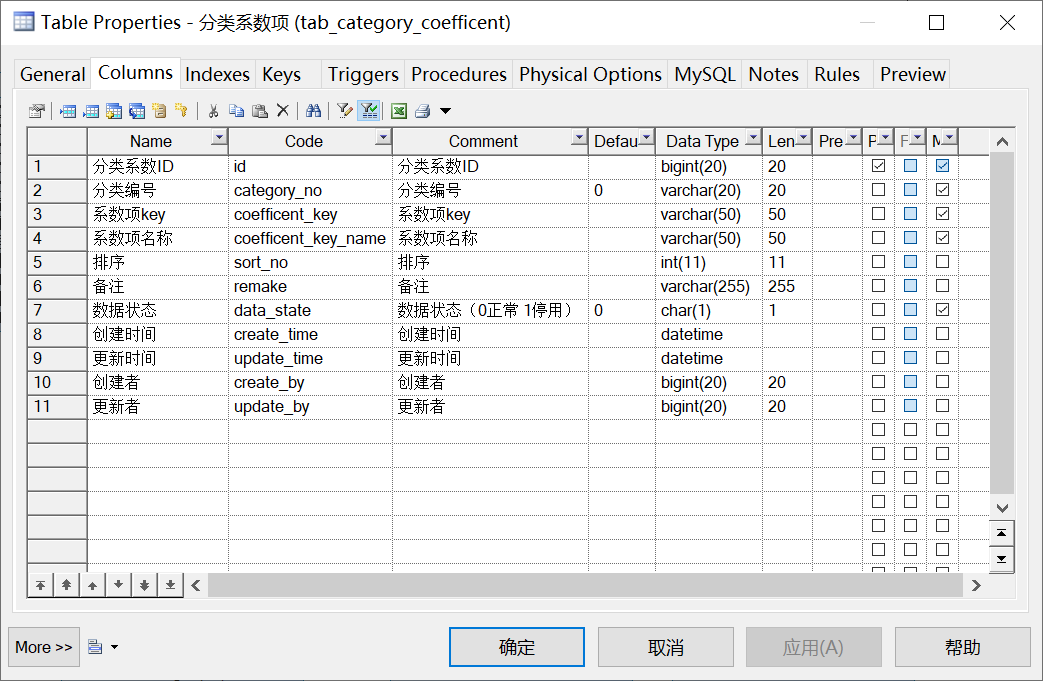
一个分类可以对应管理多个不同的系数项 ===》一对多

一个系数项可以被不同的分类关联 ===》一对多

双向一对多；**说明分类与系数项的关系为：多对多** ===》需要设计中间表tab\_category\_coefficent来保存 分类与系数的关系



与分类的关系如上图；具体的表结构如下：



* **分类与筛选**

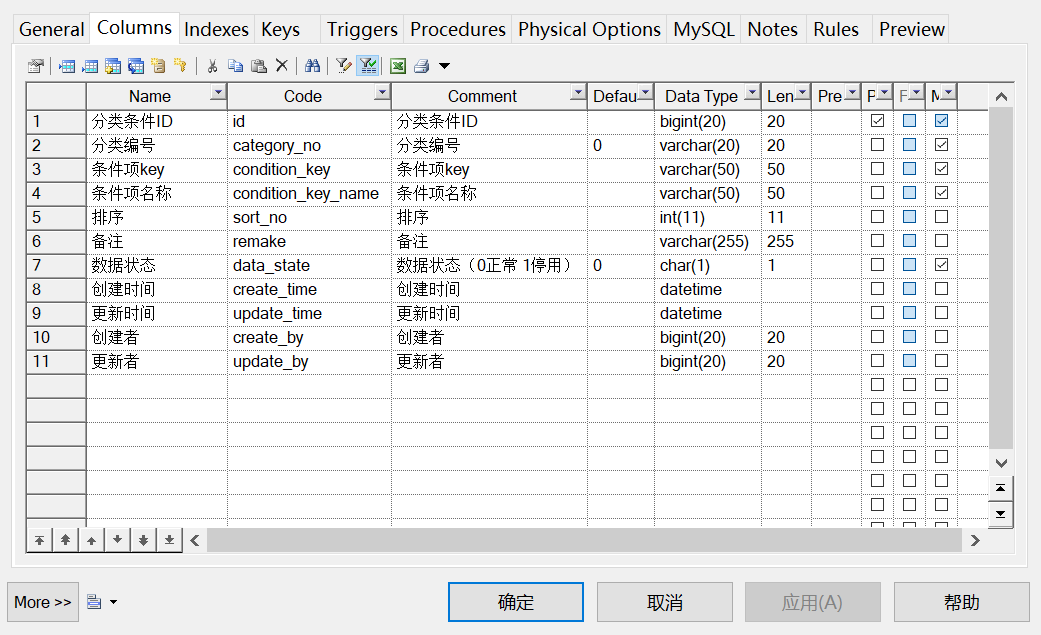
一个分类可以对应管理多个不同的筛选项 ===》一对多

一个筛选项可以被不同的分类关联 ===》一对多

双向一对多；**说明分类与筛选项的关系为：多对多** ===》需要设计中间表tab\_category\_condition来保存 分类与筛选的关系



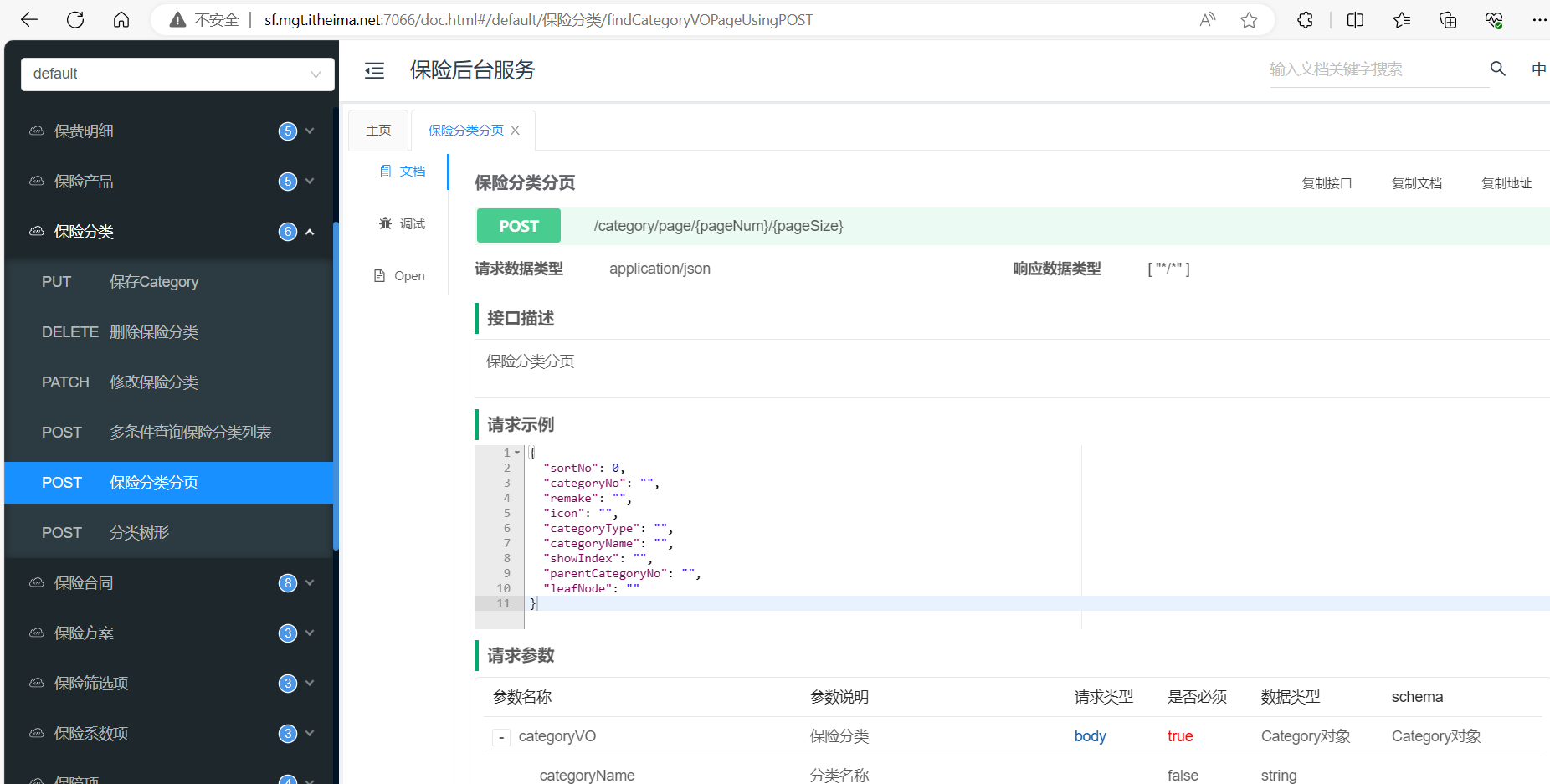
与分类的关系如上图；具体的表结构如下：



**5.3、接口设计**

**5.3.1、分页查询**

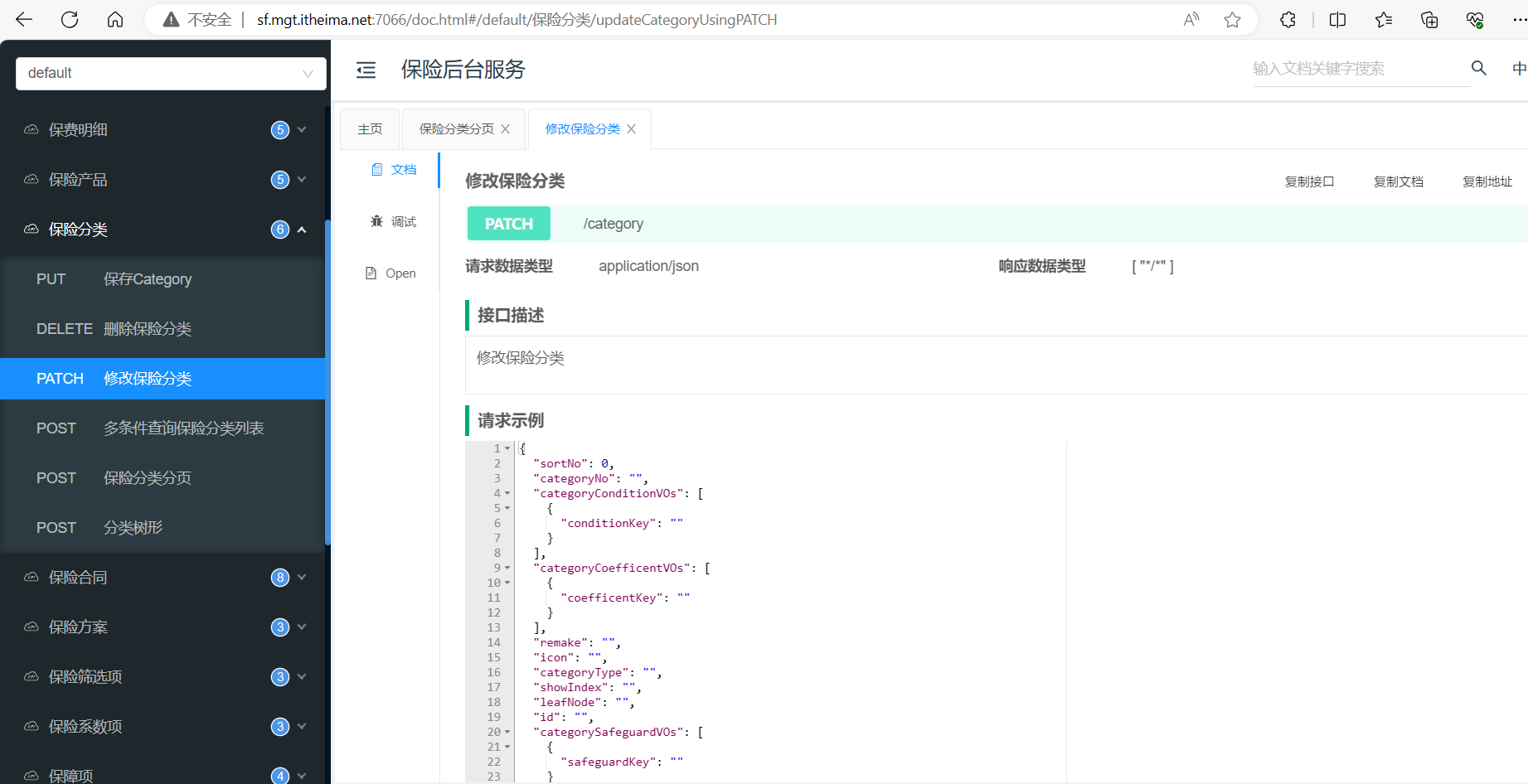
在分类的分页查询中，与前面的几个分页查询稍微有不同，它有保障项、系数项、筛选项的关联关系并且需要在查询时候**一次性全部查询出来**。接口信息地址：http://sf.mgt.itheima.net:7066/doc.html#/default/%E4%BF%9D%E9%99%A9%E5%88%86%E7%B1%BB/findCategoryVOPageUsingPOST



**5.3.2、修改--关联关系**

分类与保障项、系数项、筛选项的关联是在列表中点击 关联关系 之后的编辑页面中；也就是 修改分类。在修改的页面中发现可以添加这几项数据，但总的来说就是保存修改。具体的接口信息地址：

http://sf.mgt.itheima.net:7066/doc.html#/default/%E4%BF%9D%E9%99%A9%E5%88%86%E7%B1%BB/updateCategoryUsingPATCH



**5.4、开发实现**

**5.4.1、分页查询**

删除 findPage 方法原有内容；重写如下代码 com.itheima.sfbx.insurance.service.impl.CategoryServiceImpl#findPage

|  |
| --- |
| Java @Override @Cacheable(value = CategoryCacheConstant.*PAGE*,key ="#pageNum+'-'+#pageSize+'-'+#categoryVO.hashCode()") public Page<CategoryVO> findPage(CategoryVO categoryVO, int pageNum, int pageSize) {  try {  *//构建分页对象* Page<Category> categoryPage = new Page<>(pageNum,pageSize);  *//构建查询条件* QueryWrapper<Category> queryWrapper = queryWrapper(categoryVO);  *//查询分页基本列表数据* Page<Category> page = page(categoryPage, queryWrapper);  if (EmptyUtil.*isNullOrEmpty*(page.getRecords())) {  return null;  }  *//转换为要返回的VO列表数据* Page<CategoryVO> resultPage = BeanConv.*toPage*(page, CategoryVO.class);   *//获取分类列表的每个分类编号放置到一个集合set中* Set<String> categoryNoSet = page.getRecords().stream().map(Category::getCategoryNo).collect(Collectors.*toSet*());   */\*处理每个分类的保障项:*  *①根据查询到的分类编号查询出分类下的保障项列表；*  *②对保障项列表根据分类分组；*  *③遍历分类列表设置对应的保障项列表*  *\*/* List<CategorySafeguardVO> categorySafeguardVOs = categorySafeguardService.findListInCategoryNo(categoryNoSet);  Map<String, List<CategorySafeguardVO>> categorySafeguardVOMap = categorySafeguardVOs.stream().  collect(Collectors.*groupingBy*(CategorySafeguardVO::getCategoryNo));  if (!EmptyUtil.*isNullOrEmpty*(resultPage.getRecords())) {  resultPage.getRecords().forEach(tmpCategoryVO -> {  tmpCategoryVO.setCategorySafeguardVOs(categorySafeguardVOMap.get(tmpCategoryVO.getCategoryNo()));  });  }  */\*处理每个分类的系数项:*  *①根据查询到的分类编号查询出分类下的系数项列表；*  *②对系数项列表根据分类分组；*  *③遍历分类列表设置对应的系数项列表*  *\*/* List<CategoryCoefficentVO> categoryCoefficentVOs = categoryCoefficentService.findListInCategoryNo(categoryNoSet);  Map<String, List<CategoryCoefficentVO>> categoryCoefficentVOMap = categoryCoefficentVOs.stream().  collect(Collectors.*groupingBy*(CategoryCoefficentVO::getCategoryNo));  if (!EmptyUtil.*isNullOrEmpty*(resultPage.getRecords())) {  resultPage.getRecords().forEach(tmpCategoryVO -> {  tmpCategoryVO.setCategoryCoefficentVOs(categoryCoefficentVOMap.get(tmpCategoryVO.getCategoryNo()));  });  }  */\*处理每个分类的筛选项:*  *①根据查询到的分类编号查询出分类下的筛选项列表；*  *②对筛选项列表根据分类分组；*  *③遍历分类列表设置对应的筛选项列表*  *\*/* List<CategoryConditionVO> categoryConditionVOs = categoryConditionService.findListInCategoryNo(categoryNoSet);  Map<String, List<CategoryConditionVO>> categoryConditionVOMap = categoryConditionVOs.stream().  collect(Collectors.*groupingBy*(CategoryConditionVO::getCategoryNo));  if (!EmptyUtil.*isNullOrEmpty*(resultPage.getRecords())) {  resultPage.getRecords().forEach(tmpCategoryVO -> {  tmpCategoryVO.setCategoryConditionVOs(categoryConditionVOMap.get(tmpCategoryVO.getCategoryNo()));  });  }   *//返回结果* return resultPage;  }catch (Exception e){  *log*.error("保险分类分页查询异常：{}", ExceptionsUtil.*getStackTraceAsString*(e));  throw new ProjectException(CategoryEnum.*PAGE\_FAIL*);  } } |

**5.4.2、修改--关联关系**

**1）完善加载保障项列表**

在 分类关联——关联关系 中；查看某个分类的保障项、系数项、筛选项时候；都需要获取这些数据的列表；而 保障项是我们从无到有写的，当时并没有写查询列表的接口；需要再实现该接口。（ 系数项、筛选项都已实现不需要再写。）

查询保障项列表接口信息

|  |  |
| --- | --- |
| 说明 | 请求信息 |
| 接口地址 | http://sf.mgt.itheima.net/api/insurance-mgt/safeguard/list |
| **请求方式** | POST |
| **请求参数** | 无 |
| **响应结果** | ResponseResult<List<SafeguardVO>> |

编写处理器方法：

com.itheima.sfbx.insurance.web.SafeguardController#safeguardList

|  |
| --- |
| Java */\*\*\**  *\* 多条件查询保障项列表*  *\* @param safeguardVO 保障项VO对象*  *\* @return List*<*SafeguardVO*>  *\*/* @PostMapping("list") @ApiOperation(value = "多条件查询保障项列表",notes = "多条件查询保障项列表") @ApiImplicitParam(name = "safeguardVO",value = "保障项VO对象",required = true,dataType = "SafeguardVO") @ApiOperationSupport(includeParameters = {"conditionVO.dataState","conditionVO.conditionKey","conditionVO.conditionKeyName","conditionVO.conditionVal","conditionVO.sortNo","conditionVO.remake"}) public ResponseResult<List<SafeguardVO>> safeguardList(@RequestBody SafeguardVO safeguardVO){  List<SafeguardVO> safeguardVOList = safeguardService.findList(safeguardVO);  return ResponseResultBuild.*successBuild*(safeguardVOList); } |

实现业务方法：

com.itheima.sfbx.insurance.service.impl.SafeguardServiceImpl#findList

|  |
| --- |
| Java @NotNull private static LambdaQueryWrapper<Safeguard> queryWrapper(SafeguardVO safeguardVO) {  LambdaQueryWrapper<Safeguard> queryWrapper = new LambdaQueryWrapper<>();  *//根据保障项编号* queryWrapper.eq(!EmptyUtil.*isNullOrEmpty*(safeguardVO.getSafeguardKey()),  Safeguard::getSafeguardKey, safeguardVO.getSafeguardKey());  *//根据保障项名称* queryWrapper.like(!EmptyUtil.*isNullOrEmpty*(safeguardVO.getSafeguardKeyName()),  Safeguard::getSafeguardKeyName, safeguardVO.getSafeguardKeyName());  *//状态* queryWrapper.eq(!EmptyUtil.*isNullOrEmpty*(safeguardVO.getDataState()),  Safeguard::getDataState, safeguardVO.getDataState());  *//按照创建时间降序排* queryWrapper.orderByDesc(Safeguard::getCreateTime);  return queryWrapper; }  @Override public List<SafeguardVO> findList(SafeguardVO safeguardVO) {  try {  LambdaQueryWrapper<Safeguard> queryWrapper = *queryWrapper*(safeguardVO);  List<Safeguard> list = list(queryWrapper);  if (!EmptyUtil.*isNullOrEmpty*(list)) {  return BeanConv.*toBeanList*(list, SafeguardVO.class);  }  } catch (Exception e) {  *log*.error("查询保障项列表失败；{}", ExceptionsUtil.*getStackTraceAsString*(e));  throw new ProjectException(SafeguardEnum.*LIST\_FAIL*);  }  return List.*of*(); } |

**2）保存 分类和关联**

在关联关系时候；就是调用修改方法；提交到后台的 CategoryVO 数据结构参考如下：

|  |
| --- |
| JSON {  "id": "1715250693712924674",  "createTime": "2023-10-20 14:16:48",  "updateTime": "2023-11-08 10:04:02",  "createBy": "1371500419615895553",  "updateBy": "1371500419615895553",  "dataState": "0",  "creator": null,  "parentCategoryNo": "100001001000000",  "categoryNo": "100001001002000",  "categoryName": "门诊险",  "icon": "",  "leafNode": null,  "showIndex": "0",  "categoryType": "1",  "sortNo": 9,  "remake": null,  "checkRule": "0",  "checkedIds": null,  "categoryConditionVOs": [  {  "conditionKey": "SX006",  "conditionKeyName": "保多久"  },  {  "conditionKey": "SX003",  "conditionKeyName": "特色保障"  }  ],  "categoryCoefficentVOs": [  {  "coefficentKey": "XS008",  "coefficentKeyName": "投入周期"  },  {  "coefficentKey": "XS001",  "coefficentKeyName": "有无医保"  }  ],  "categorySafeguardVOs": [  {  "safeguardKey": "BZ042",  "safeguardKeyName": "专家门诊预约"  },  {  "safeguardKey": "BZ001",  "safeguardKeyName": "一般医疗保险金"  }  ],  "checkedCategoryNos": null,  "nodeFloors": null } |

分析上述的数据结构；可以发现包含了：分类基本信息、分类对应的保障项、系数项、筛选项集合；也就是说；需要同时将这4类数据保存到数据库中就算实现完成。

具体实现代码：

com.itheima.sfbx.insurance.service.impl.CategoryServiceImpl#update

|  |
| --- |
| Java @Override @Transactional @Caching(evict = {@CacheEvict(value = CategoryCacheConstant.*PAGE*,allEntries = true),  @CacheEvict(value = CategoryCacheConstant.*LIST*,allEntries = true),  @CacheEvict(value = CategoryCacheConstant.*BASIC*,key = "#categoryVO.id")}) public Boolean update(CategoryVO categoryVO) {  try {  *//1、保存分类基本信息* Category category = BeanConv.*toBean*(categoryVO, Category.class);  boolean flag = updateById(category);   *//2、保存分类保障项信息*  *//2.1、删除分类保障项：根据分类编号删除分类保障项* categorySafeguardService.deleteByCategoryNo(category.getCategoryNo());  *//2.2、保存分类保障项：保存每个分类保障项需要设置分类编号* List<CategorySafeguard> categorySafeguardList = categoryVO.getCategorySafeguardVOs().stream().map(tmpCategorySafeguardVO -> {  tmpCategorySafeguardVO.setCategoryNo(category.getCategoryNo());  return BeanConv.*toBean*(tmpCategorySafeguardVO, CategorySafeguard.class);  }).collect(Collectors.*toList*());  categorySafeguardService.saveBatch(categorySafeguardList);   *//3、保存分类系数项信息*  *//3.1、删除分类系数项：根据分类编号删除分类系数项* categoryCoefficentService.deleteByCategoryNo(category.getCategoryNo());  *//3.2、保存分类系数项：保存每个分类系数项需要设置分类编号* List<CategoryCoefficent> categoryCoefficentList = categoryVO.getCategoryCoefficentVOs().stream().map(tmpCategoryCoefficentVO -> {  tmpCategoryCoefficentVO.setCategoryNo(category.getCategoryNo());  return BeanConv.*toBean*(tmpCategoryCoefficentVO, CategoryCoefficent.class);  }).collect(Collectors.*toList*());  categoryCoefficentService.saveBatch(categoryCoefficentList);   *//4、保存分类筛选项信息*  *//4.1、删除分类筛选项：根据分类编号删除分类筛选项* categoryConditionService.deleteByCategoryNo(category.getCategoryNo());  *//4.2、保存分类筛选项：保存每个分类筛选项需要设置分类编号* List<CategoryCondition> categoryConditionList = categoryVO.getCategoryConditionVOs().stream().map(tmpCategoryConditionVO -> {  tmpCategoryConditionVO.setCategoryNo(category.getCategoryNo());  return BeanConv.*toBean*(tmpCategoryConditionVO, CategoryCondition.class);  }).collect(Collectors.*toList*());  categoryConditionService.saveBatch(categoryConditionList);   return flag;  }catch (Exception e){  *log*.error("修改保险分类异常：{}", ExceptionsUtil.*getStackTraceAsString*(e));  throw new ProjectException(CategoryEnum.*UPDATE\_FAIL*);  } } |

**6、午间问题**

* **说出常见保险保障项**
* 一般医疗保险金
* 特定疾病医疗保险金
* 重大疾病医疗保险金
* 重大疾病关爱保险金
* 特定药品费用医疗保险金
* 门诊挂号及陪诊、100种重大疾病津贴保险金
* 押金垫付、癌症质子重离子放疗、恶性肿瘤特效药、就医绿色通道、医疗费用垫付
* **常见系数项**
* 有无医保
* 投入方式
* 付款方式
* 年龄范围
* 保障期限
* 投入时长
* 保障金额
* 自动续保
* 投入周期
* 什么时候开始领
* 领取多久
* 领取周期
* **常见筛选项**
* 为谁买
* 健康状况
* 特色保障
* 特色服务
* 保多久
* 交通工具
* **常见分类**

医疗险、重疾险、意外（出行、运动）、旅游（境内、境外）、人寿（终身、定期、双全）、年金（养老、教育、财富）

* **你工作时的开发流程描述一下**
* **你日常开发都使用哪些软件工具**
* **说一下你开发过最复杂的业务涉及多少张表？分别是什么？涉及最多字段的表有几个字段？**
* **遇到这种关联性、关系比较多和复杂的表；你有什么设计或者优化经验吗**
* **简历职责-候选**：优化保险管理后端中分类列表查询的查询效率；将分类、系数、保障项、筛选项等7张表的连接查询从335ms优化到150ms，查询效率提升超过一倍。

下次课====》进行设计实战

* 新需求
* 保险基础数据