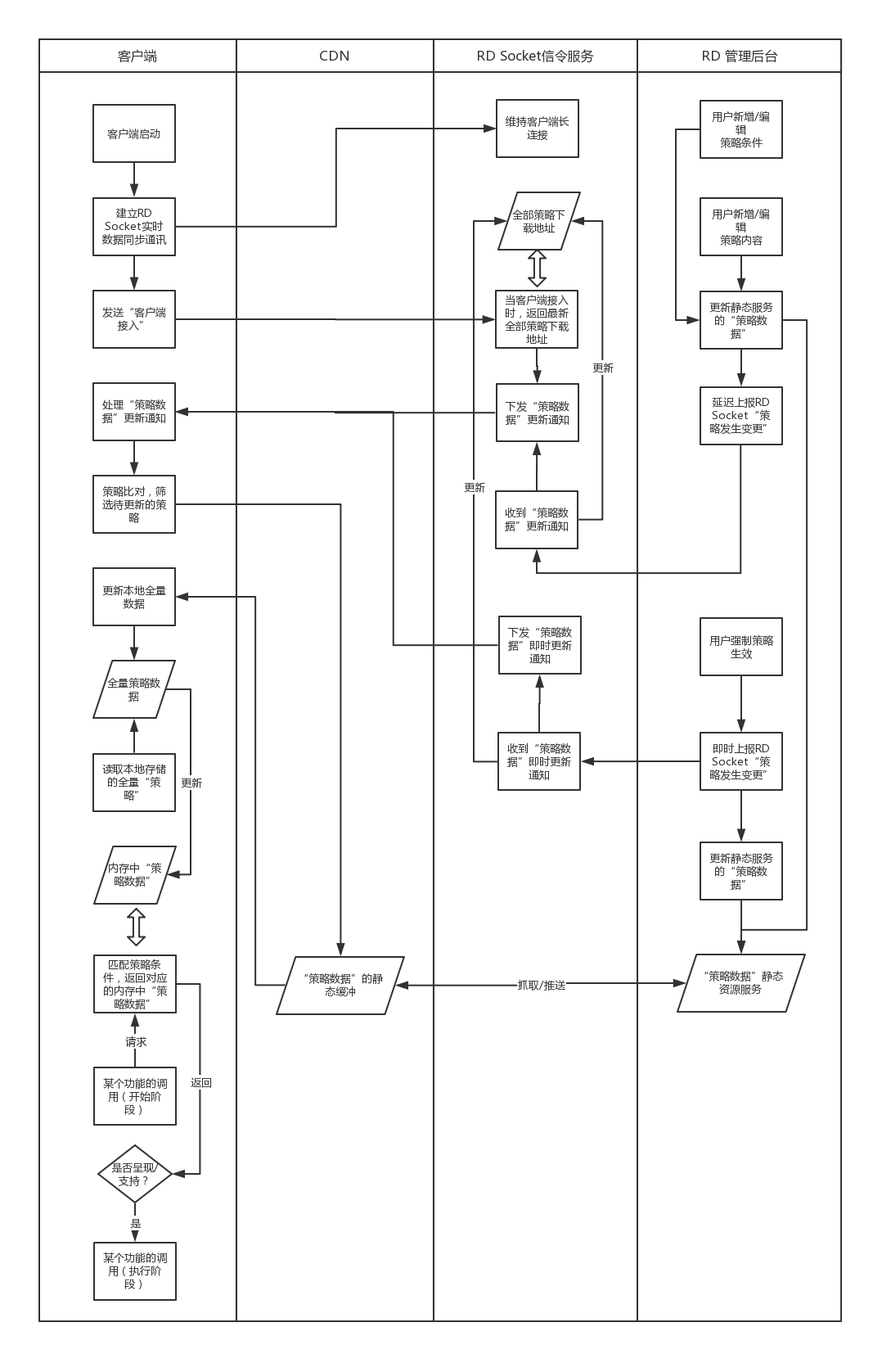
# RemoteDecition远程决策管理后台

# Beta 1.0 新手引导

# 前序

RemoteDecition远程决策系统，用于线上远程控制各个接入端动态即时开关某些功能以及改变显示效果。可以用于临时屏蔽**无法进行热更新的各个接入端**的线上紧急问题

# 实现思路



* “策略数据”指某一项功能对应的相关配置信息，其内容格式为JSON，例如（范例一）：

{

"supportScreenshots":"1",/\*是否支持截屏功能\*/

"enableScrrenshots":"1",/\*是否默认开启截屏功\*/

"defaultPagePerSecond":"1",/\*每秒截屏帧数\*/

"defaultSnapshotsHeight":"960",/\*默认生成截图尺寸的高度\*/

"defaultSnapshotsWidth":"1280"/\*默认生成截图尺寸的宽度\*/

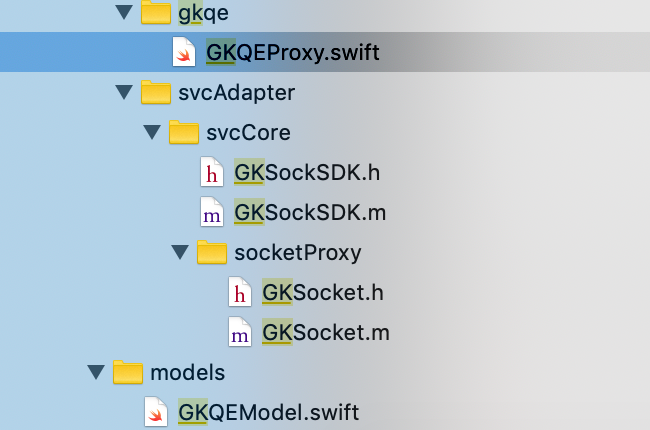
}

* 每一个策略都配以一系列的匹配条件,用来与客户端发送请求中的各种参数进行匹配，最终确定并返回一条最适合当前客户端的某项功能的“策略数据”。
* 客户端的“策略数据”与服务端的“策略数据”保持实时同步，每一条“策略数据”被保存为一个独立的文件（由SDK自动管理，不需要客户端研发维护）
* 客户端在启用某项功能时会向本地SDK发送请求，获取（无需等待，即时获取）一条与当前功能相匹配的“策略数据”，用来开、关功能，或做出相对应的响应。

# 使用方法

## 客户端使用方法

### 第1步，项目中集成RemoteDecition SDK或导入源文件



其中GKQEProxy.swift中定义了所有操作“策略数据”的公有接口，以供客户端使用。定义如下：

/\*\*

GKQEProxy的静态,所有“策略数据”相关的请求，均通过该单例进行访问，不允许自行初始化GKQEProxy

\*/

+(GKQEProxy \*)instance

/\*\*

启动SDK，对象方法。

\*/

-(void)start

/\*\*

停止SDK，对象方法。

\*/

-(void)stop

/\*\*

设置SDK初始化成功后的回调函数，参数initCompleteCallBack 是一个无参数、无返回值的函数。该方法建议在start函数调用前调用，否则initCompleteCallBack 可能不会被调用

\*/

-(void)setInitEndCallBack(initCompleteCallBack ()->void)

/\*\*

获取某一个功能模块对应的“策略配置”数据,参数strategyCategroyName代表策略名，reqArgs是请求参数 采用key，value形式组装，key必须为String类型，value可以为任何基本数据类型String、Int、Bool、 Double、 Float

\*/

-(void)requestGKQE(strategyCategroyName String,reqArgs Map<String,AnyObject>)

/\*\*

获取某一个功能模块对应的“策略配置”数据中的指定字段对应的值

\*/

-(GKQEValue \*)getGKQE(key String)

/\*\*

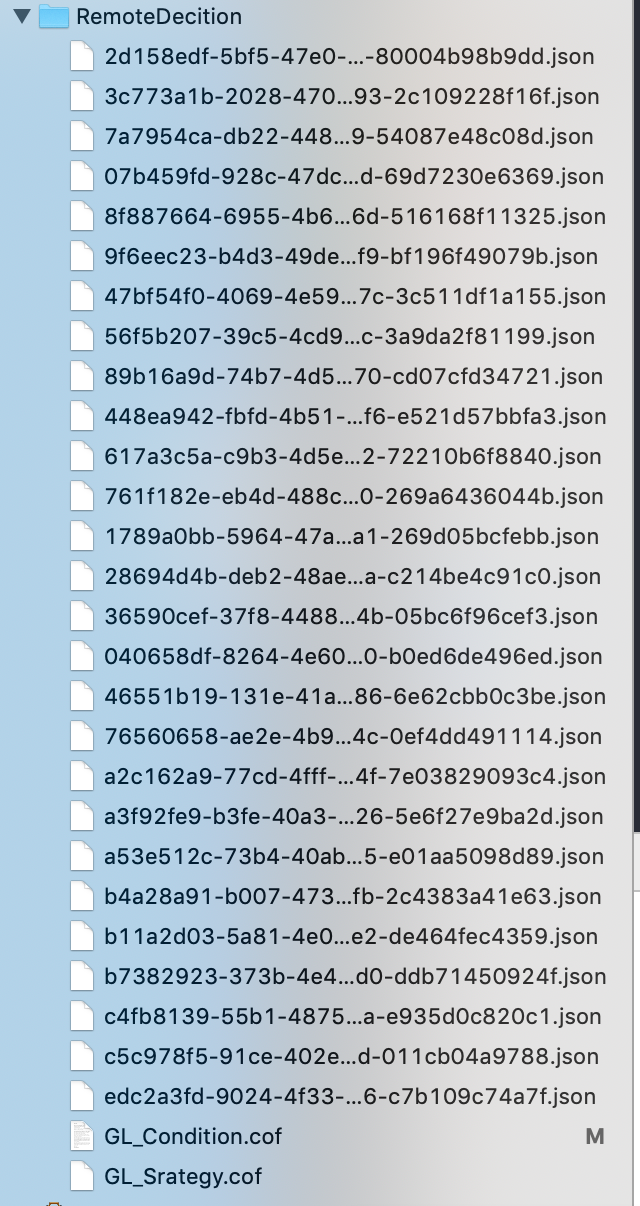
该方法慎用

向“策略配置”数据中的指定字段写入值。注意该值只会临时生效，待程序重启后将重新使用本地缓存的“策略配置”中的数据。

\*/

-(void)setGKQE(key String,val AnyObject)

### 第2步，项目中引入默认策略配置文件夹（主要用于第一次启动，未能成功连接RD Socket服务的用户）



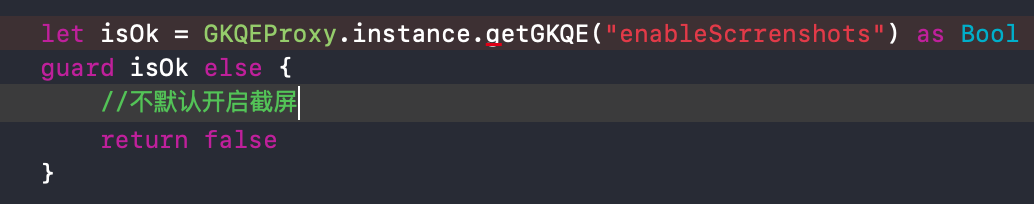
其中有两个关键文件，GL\_Condition.cof与GL\_Strategy.cof，之后再详细介绍

### 第3步，在项目源码中的，程序启动入口位置调用GKQProxy.instance.start（）用来启动SDK

### 第4步，在需要用到策略数据时（如启动某个功能模块时），调用GKQProxy.instance.requestGKQE（“策略名”，”参数”）,用与将本地文件中的策略数据加载到内存中，以供后续使用，被加载的数据,其格式同范例一, 调用实例如下：



### 第5步，在代码关键位置调用GKQProxy.instance.getGKQE(“键值”),获取指定的策略值，用于功能开关与呈现样式。示例如下：



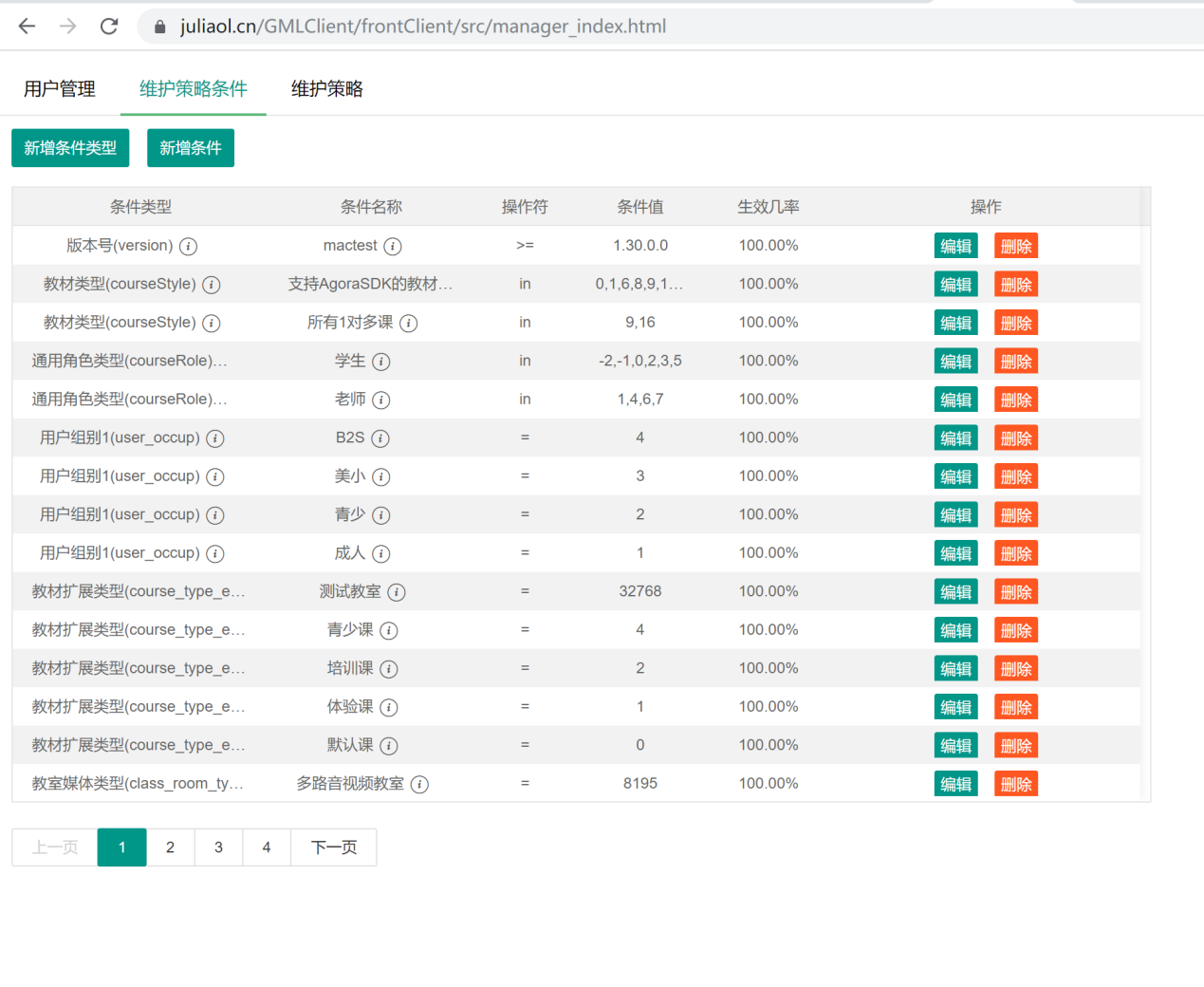
### So, easy !!

## RD管理后台使用方法

### 第1步,登录RD管理后台，临时地址（暂无密码）：

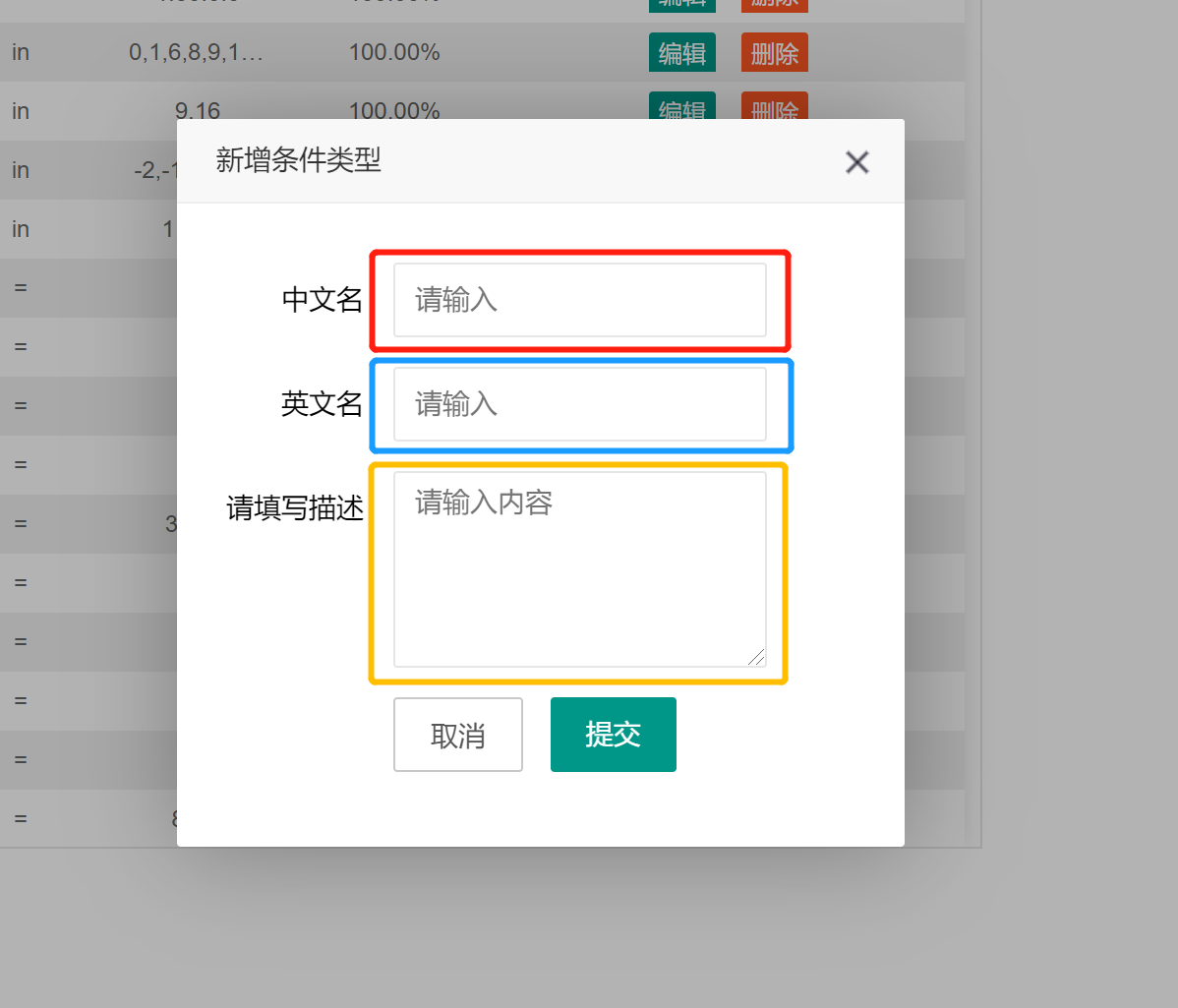
<https://www.juliaol.cn/GMLClient/frontClient/src/manager_index.html>

界面如下



### 第2步，新建条件类型





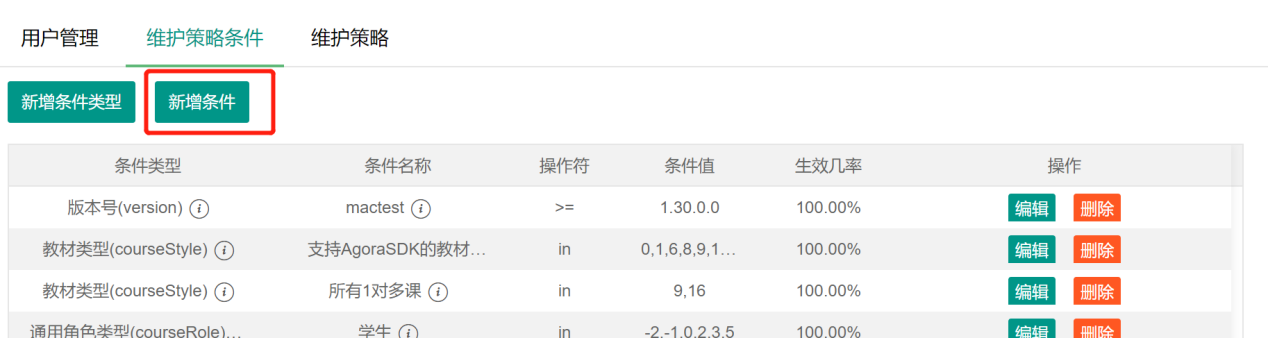
红框内，填写 条件类型的中文名，方便之后使用时容易理解该条件是做什么用的

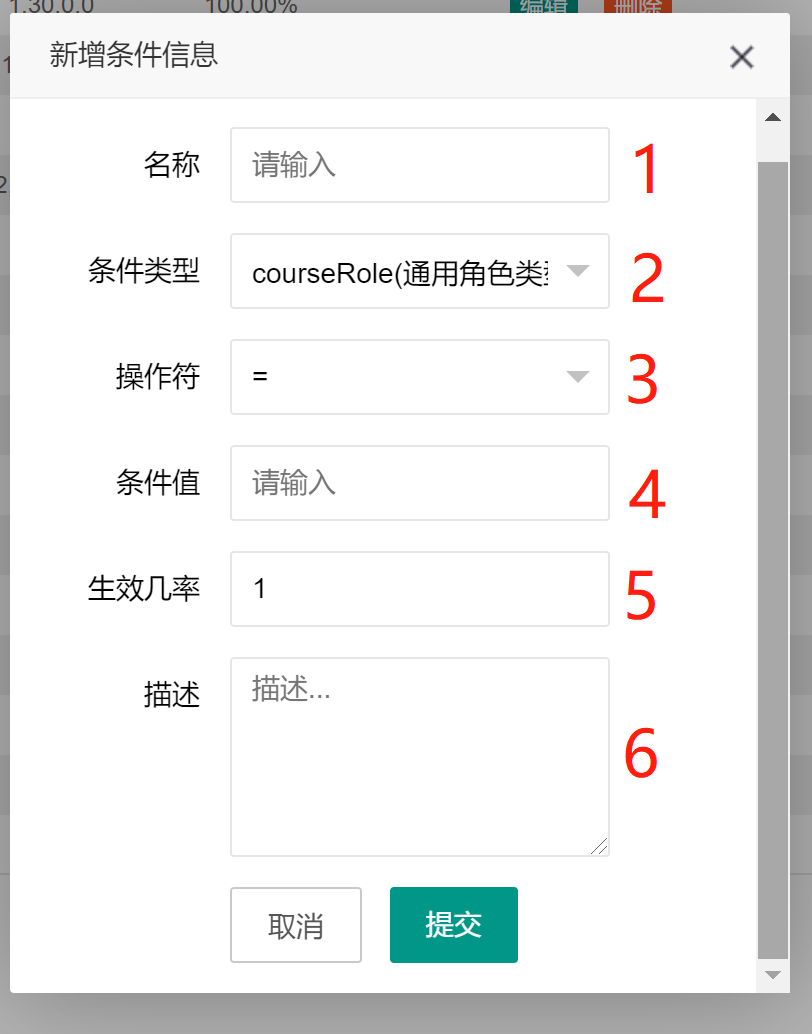
黄框内，填写 该条件类型的详细描述，非必填。

蓝框内，填写 条件类型的英文名，即对应客户端请求时的参数中的key（下图绿色框内的部分），严格区分大小写，非常重要



### 第3步, 新增条件





文本框1中，填写条件的名称，如：1v1老师、学生、管理员、公开课老师、PSO课老师，用来描述条件，方便查看。

文本框2中，选择条件的类型，即上一步刚刚创建的条件类型。

文本框3中，填写该条件的匹配算法，如：‘=’代表 当客户端请求参数中的某个key与对应的value 完全与条件类型和条件值完全相同时，该条件视为匹配成功。

文本框4中，填写条件的值。

文本框5中，填写条件的生效几率，范围在0 - 1之间。当该条件与客户端请求参数匹配成功后，会对生效几率进行运算，1代表100%生效，0代表一定不生效。

文本框6中， 填写条件的描述，无实际意义，非必填。

### 第4步，新建策略类别，不要认为策略是单独的，客户端请求指定测略时所发送的“策略类别名称”与“策略参数”就是在一组策略中匹配一个最适合的策略），比如策略类别“1对1教室策略”中有可能有“老师对应的策略”、“学生对应的策略”,“监课cc用户对应的策略”等等。见下图：



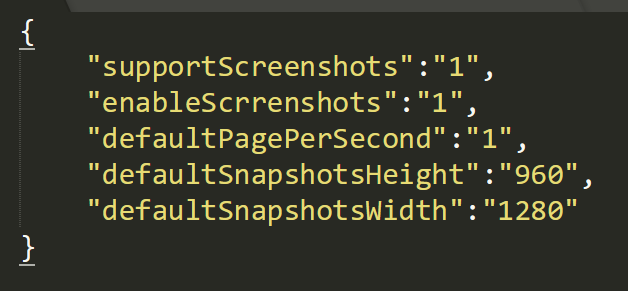


文本框1中，填写 策略类别名称， 对应客户端请求时传的strategyCategroyName。

文本框3中，填写 策略相关描述，非必填。

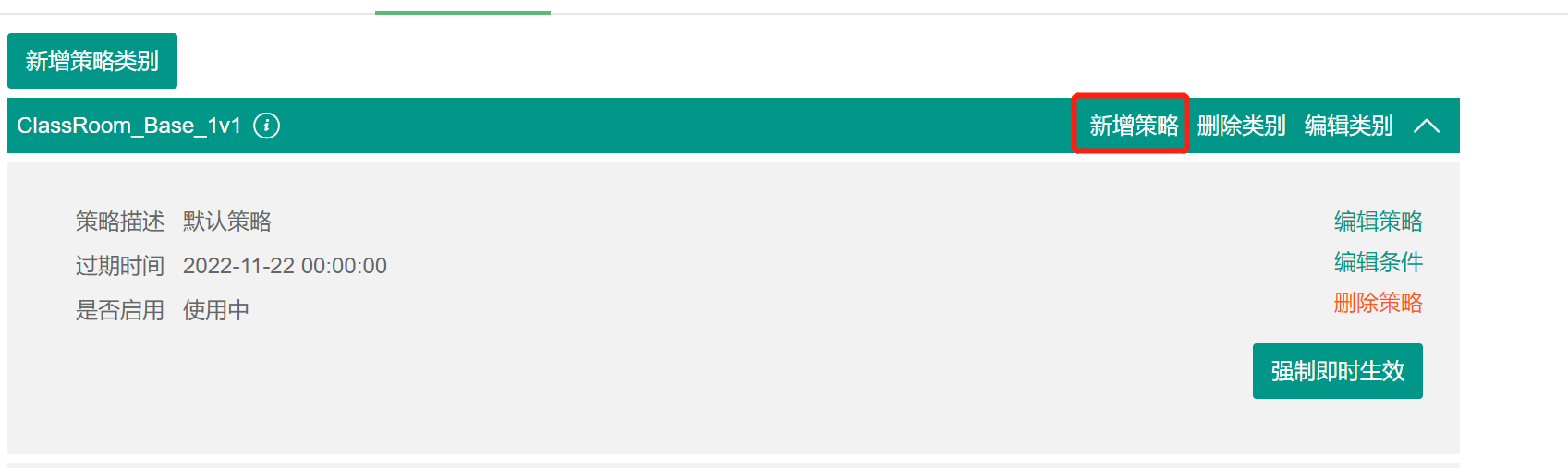
文本框2中，填写 “策略模板”，注意“策略模板”要与“策略数据”区分开。如下：

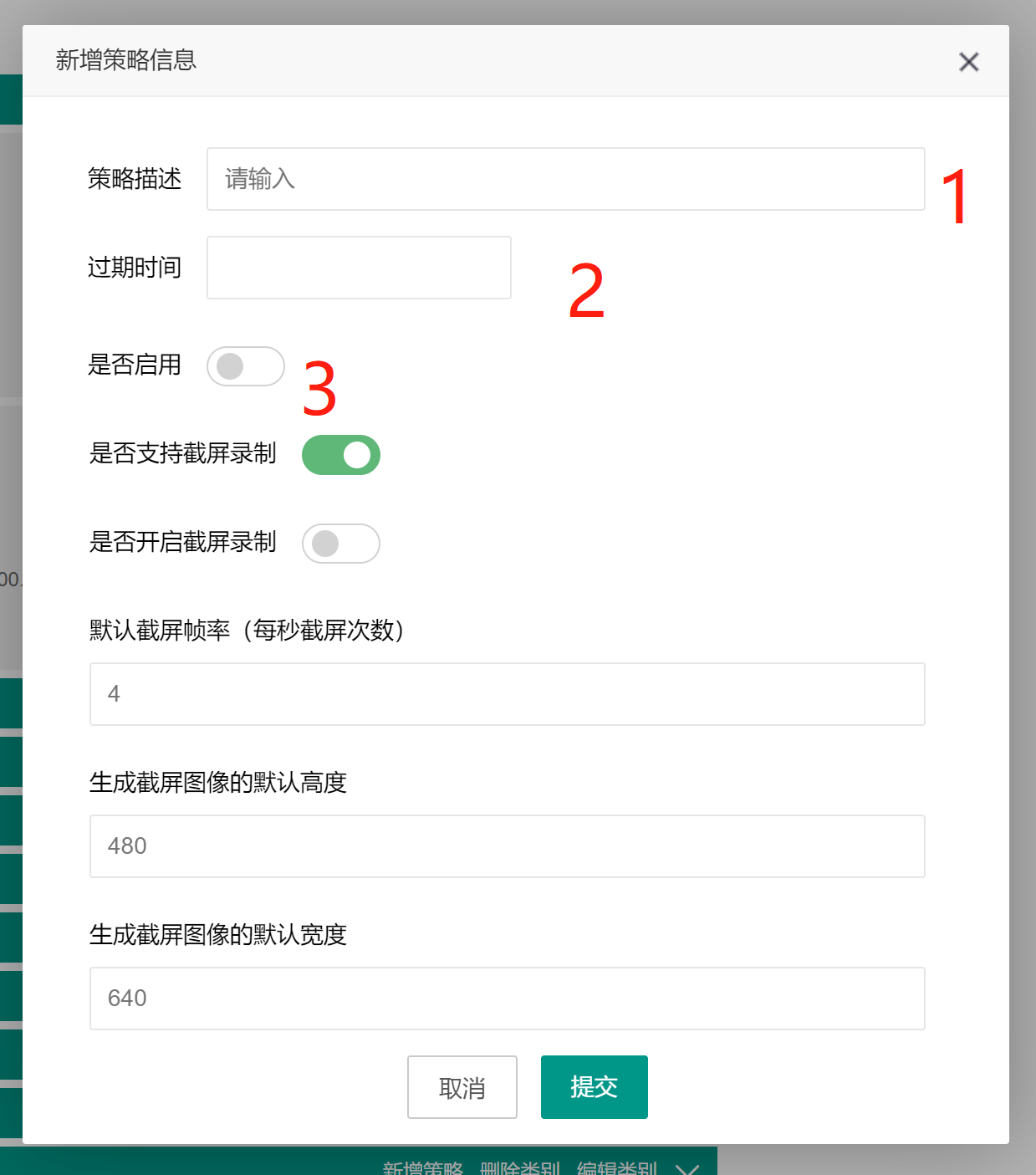
这是“策略模板”

这是“策略数据”

“策略模板”的作用是让RD管理后台能够动态生成表单界面，已供用户填写内容，并做数据格式急眼。当用户点击提交后，RD管理后台还会应用“策略模板”的格式组装用户数据，最终生成客户端使用的“策略数据”。（PS：如果不会写策略模板的话，可以让后台管理员帮忙编写）。

### 第5步，新建策略。





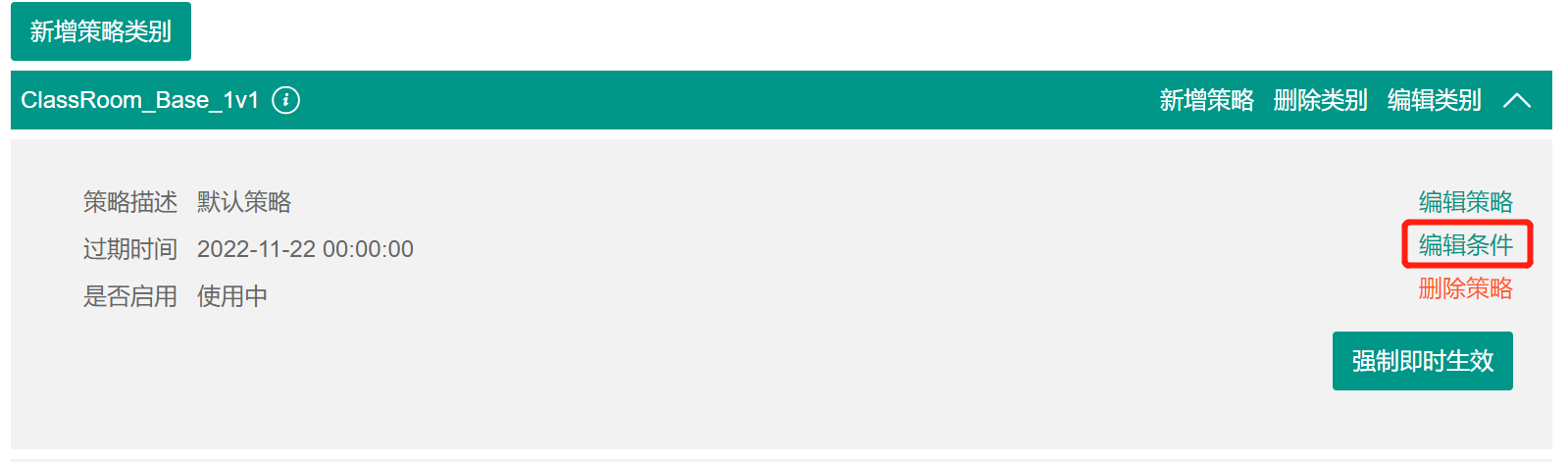
文本框1中，填写 该策略的相关描述，比如“1v1课中的老师策略”

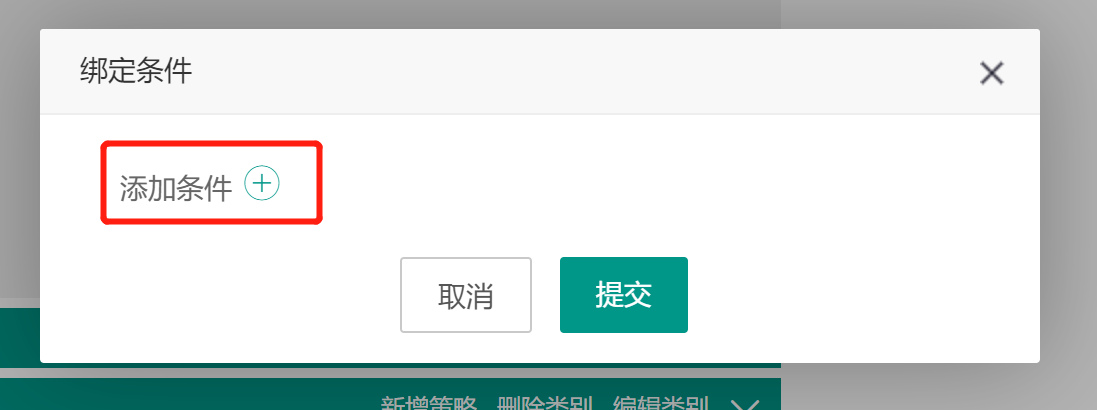
文本框2是一个时间选择器，可以选择该策略的失效时间

开关3用来控制当前策略是否启用，默认为不启用。如果策略不启用，则不会在客户端匹配策略时进行匹配计算。

其它文本框为用户自定义字段，根据业务方需求填写。

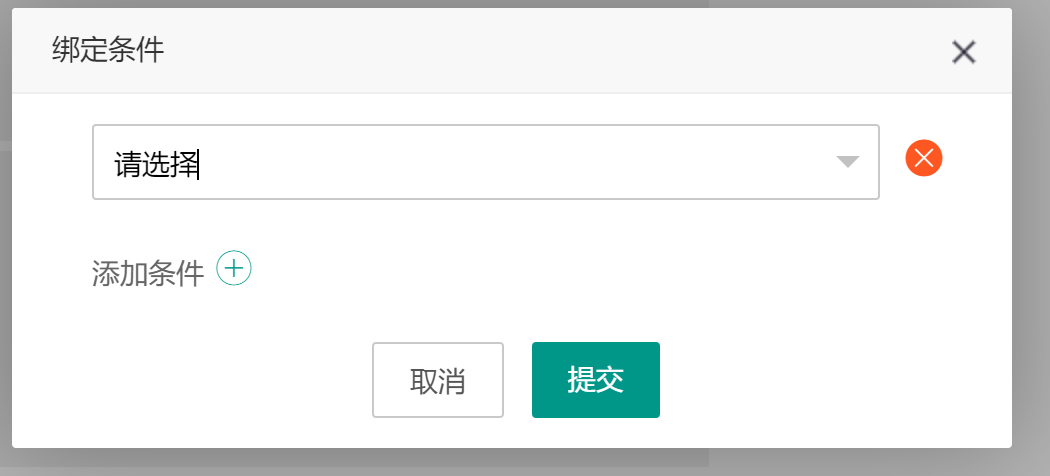
### 第6步，为策略绑定条件。



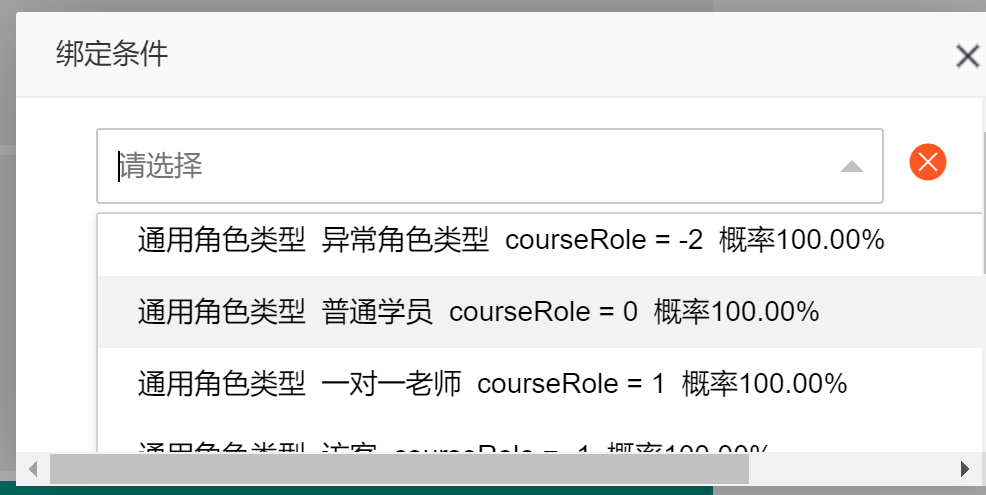


点击加号，可以新增条件。如果策略不绑定任何条件，则当客户端请求指定策略数据时，假设所有其它策略都没有匹配成功，会默认选择这条策略 作为生效策略。

点击加号后会显示：



这是一个支持模糊搜索功能的下拉框



条件绑定成功后，效果如下：



### 完成。

### 对上述流程的简要归纳：

### 登录网站RD管理后台

### 新建策略条件类别

### 新建策略条件

### 新建策略类别

### 新建策略

### 为策略绑定条件

### 结束，So, easy !!

# 答疑环节