



D2 前端技术论坛
D2 FRONTEND TECHNOLOGY FORUM

精心

第十六届 D2 前端技术论坛

面向中后台复杂场景的低代码实践思路

偏左



Contents

目录

01 方案背景

02 动态标注生成交互界面

03 策略编排生成逻辑代码

04 Next

01

方案背景

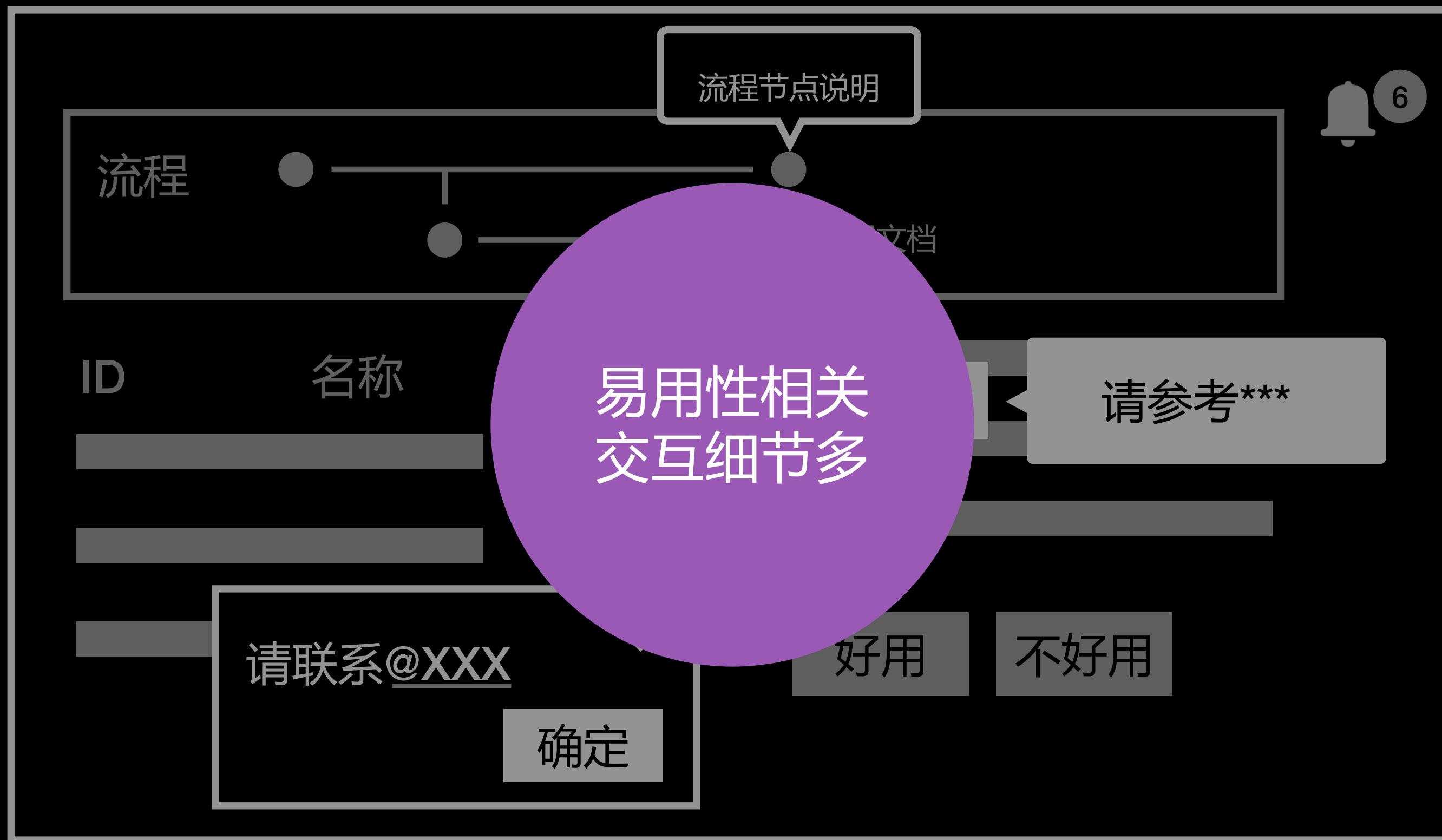
复杂视图 / 复杂逻辑问题



理不清的业务规则，逻辑逐渐杂乱

看不懂改不动，幻想if/else搞定一切

- 业务规则由多个责任人掌握
- 业务场景多变化快，迭代频繁
- 业务固有复杂度以及历史包袱



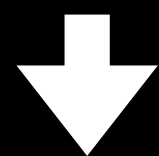
提示指引等易用性交互将开发复杂化

复制/粘贴加速腐化，容易干扰业务功能

- 内容运营需要解释的内容多
- 易用性交互形式多，并不限于图文
- 易用性需求排期低于功能需求，平台越来越难用

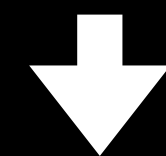
你中招了吗？

复杂视图



动态标注

复杂逻辑



策略编排



D2 前端技术论坛
D2 FRONTEND TECHNOLOGY FORUM

精心

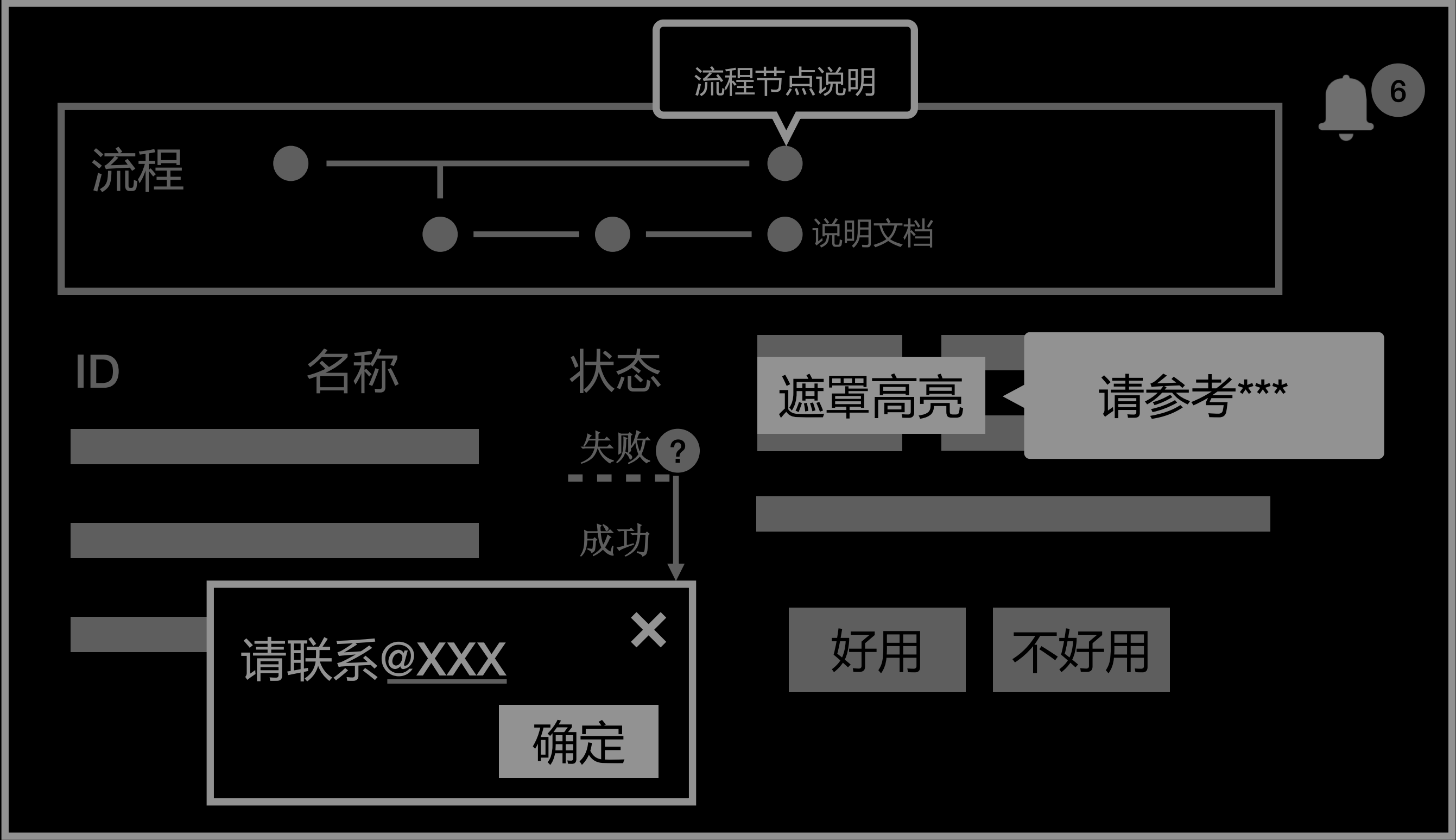
02

动态标注生成交互界面

面向复杂交互视图的方案

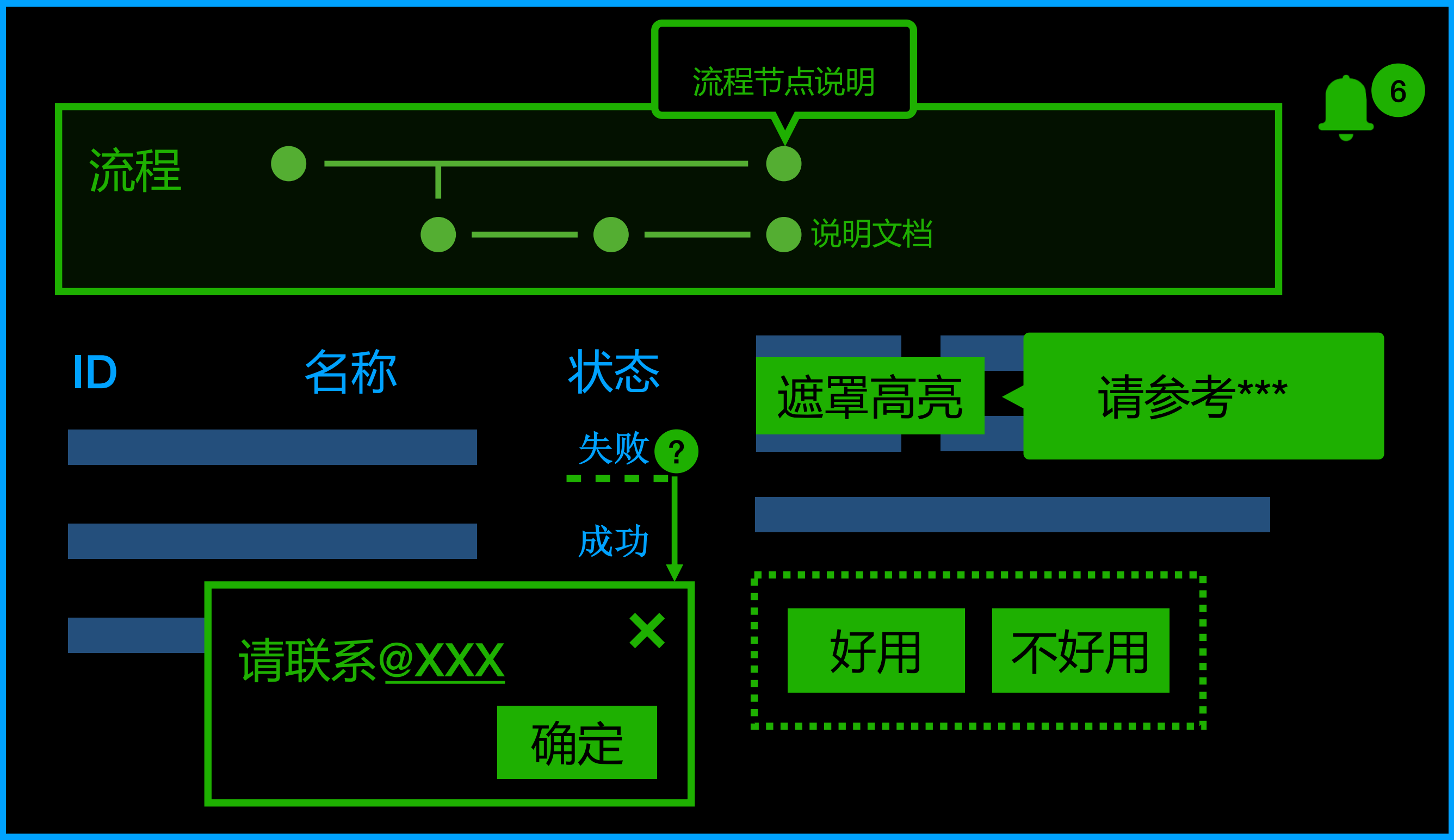
中后台页面示例

■ 典型中后台交互



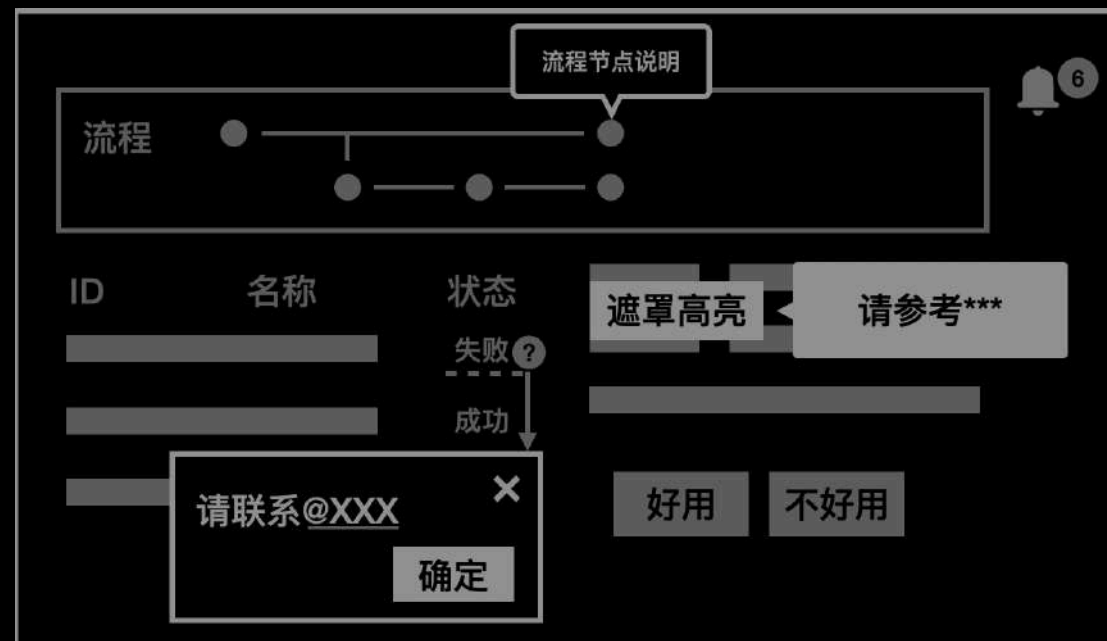
页面按功能划分

■ 业务功能交互 ■ 辅助内容交互



方案核心目标

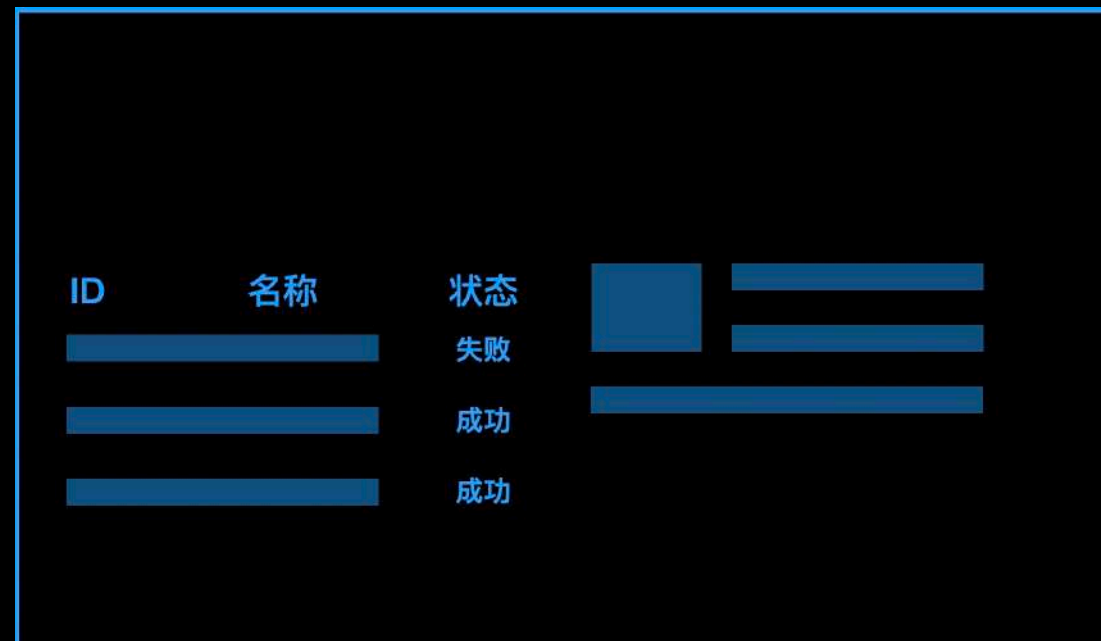
所有类型交互



ProCode开发

面向：前端

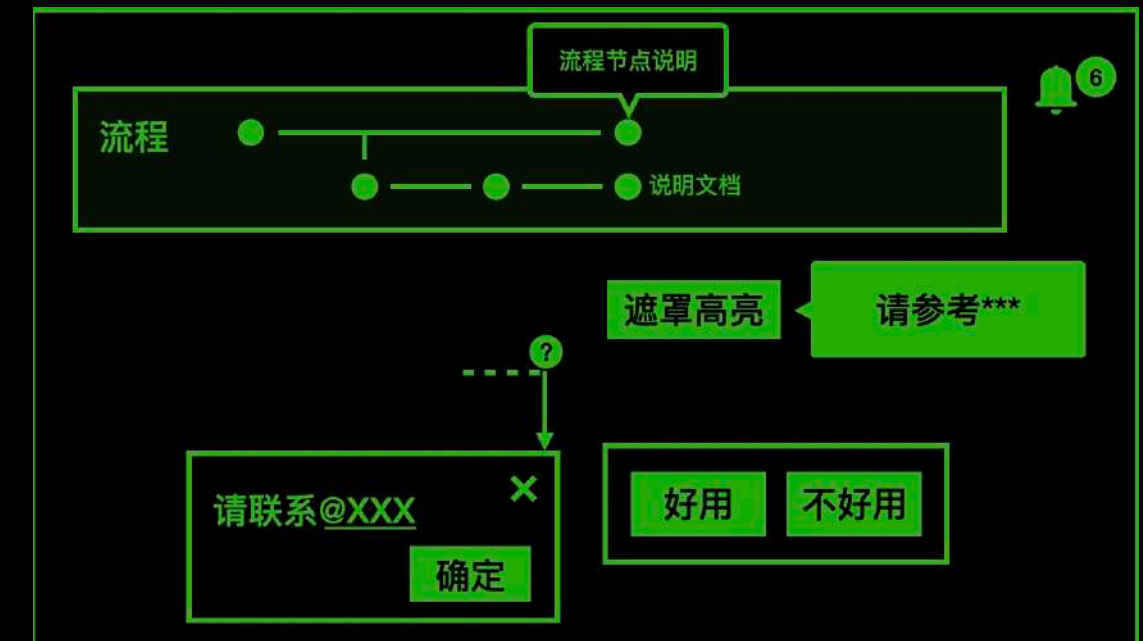
业务功能交互



专注ProCode开发

面向：前端

辅助内容交互



低/无代码配置

面向：运营 / UED / PD

方案实施效果

需要加提示说明 点击出现弹窗说明

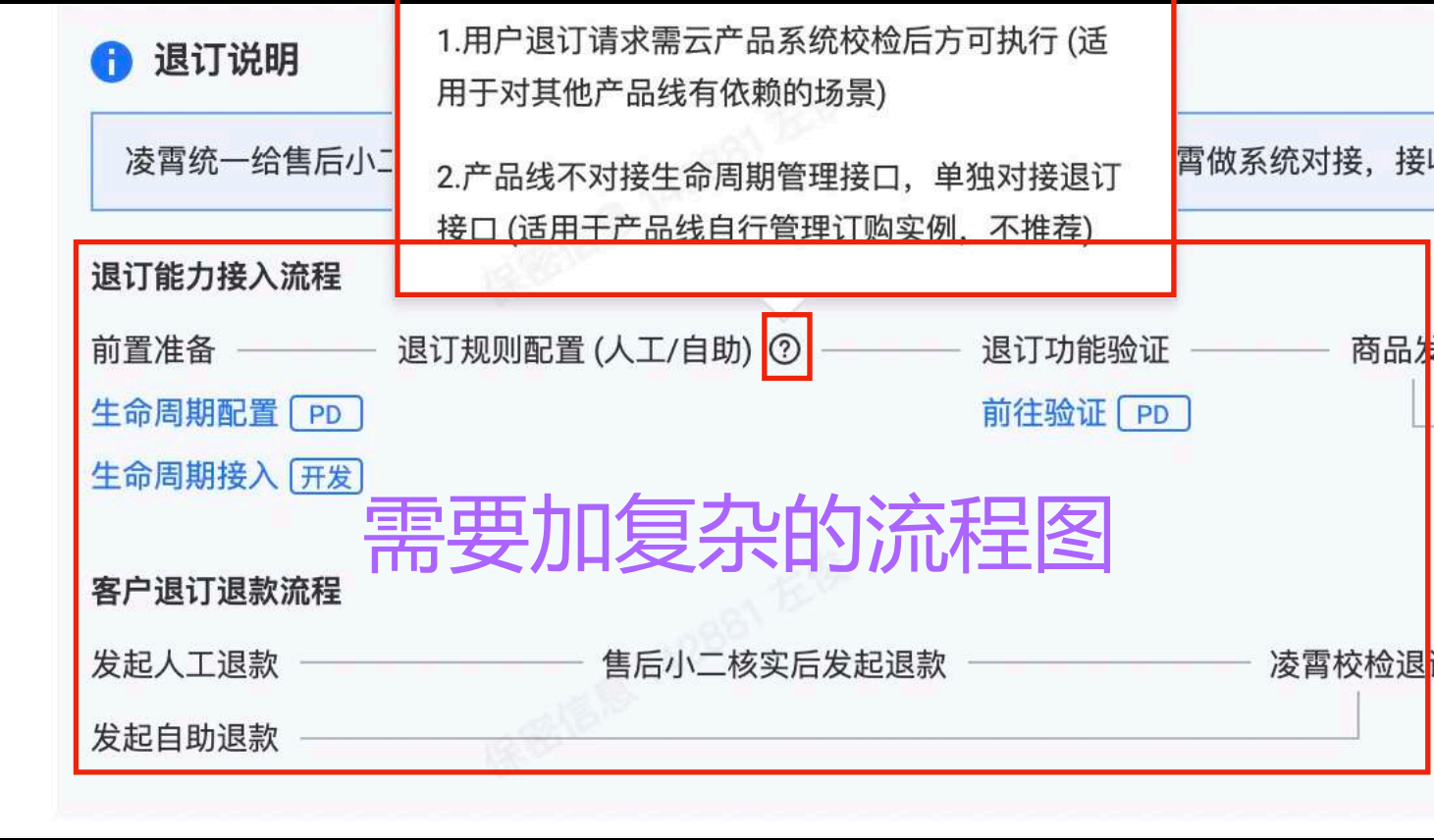


Step引导

展示相关解释

点击出现文档

流程说明中需要加气泡提示



PT播放

需要加复杂的流程图

识别出关键词或术语

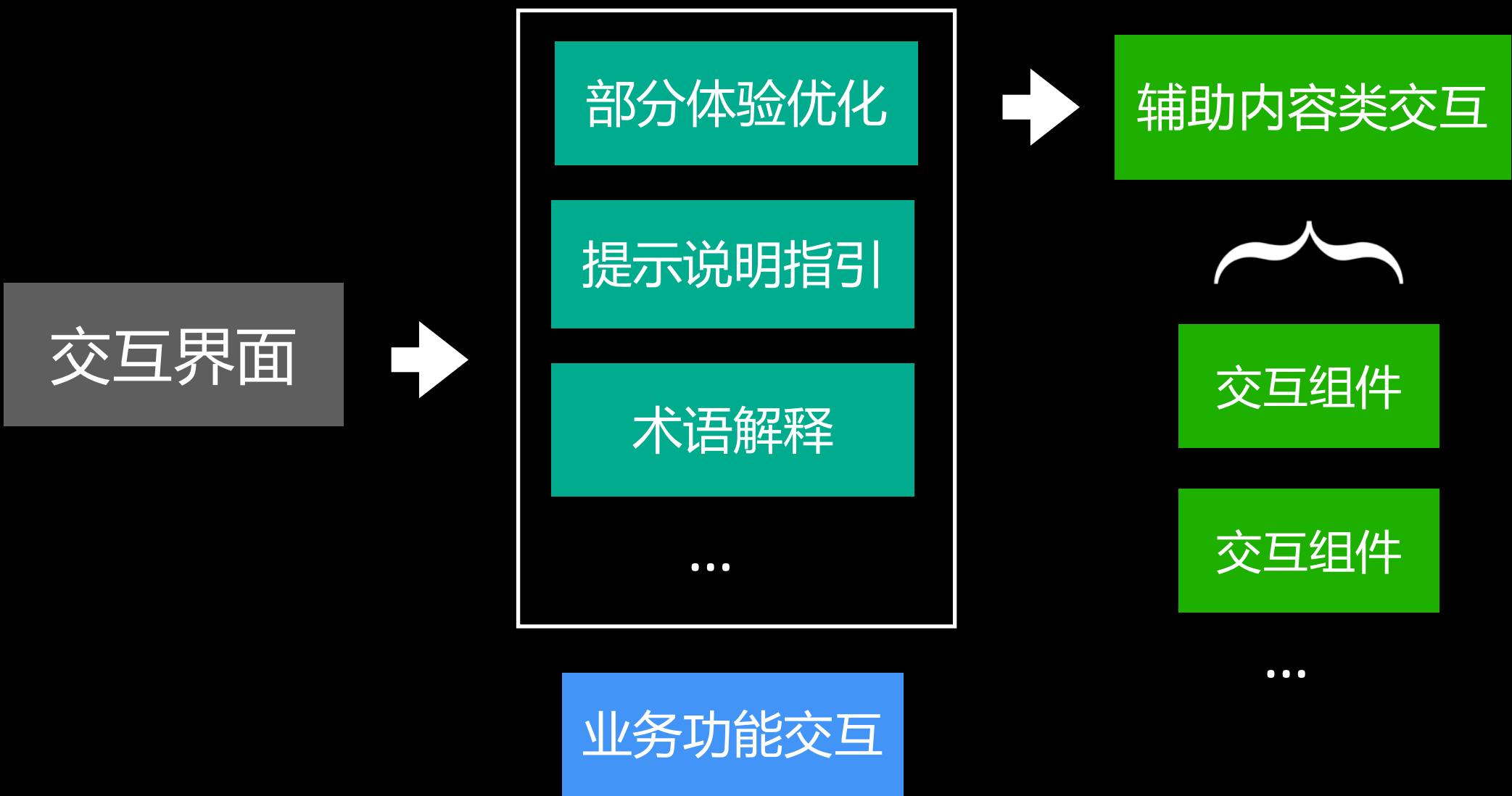
几乎所有的辅助类交互近400个，都通过配置完成

动态标注方案实践

Step 1

分解交互界面，找到辅助类交互

将辅助类交互分解成各个组件，分别配置展示样式、内容等等



- 容器类 通用(div) / 弹窗 / 消息条 / 气泡 / 通知
- 内容类 按钮 / 文本 / 图标 / 图片 / SVG / Markdown
- 场景类 步骤向导 / 关键词 / 视频播放 / PPT播放 ...
- 外部能力 反馈调研 / 工单 / 知识库 / 机器人问答 ...

StepByStep引导

配置化提示消息

嵌入式文档

嵌入式PPT / PDF

引导面板

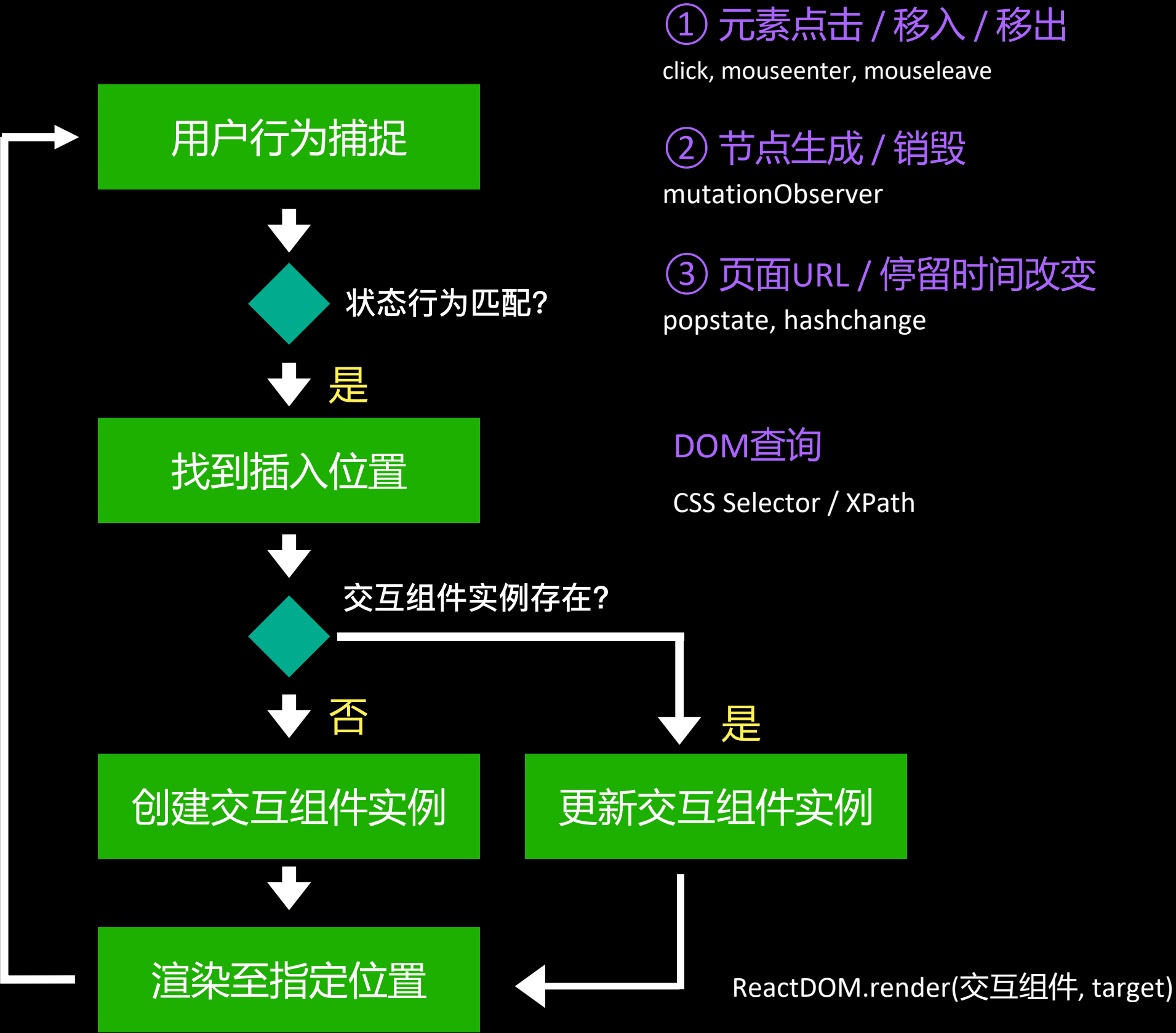
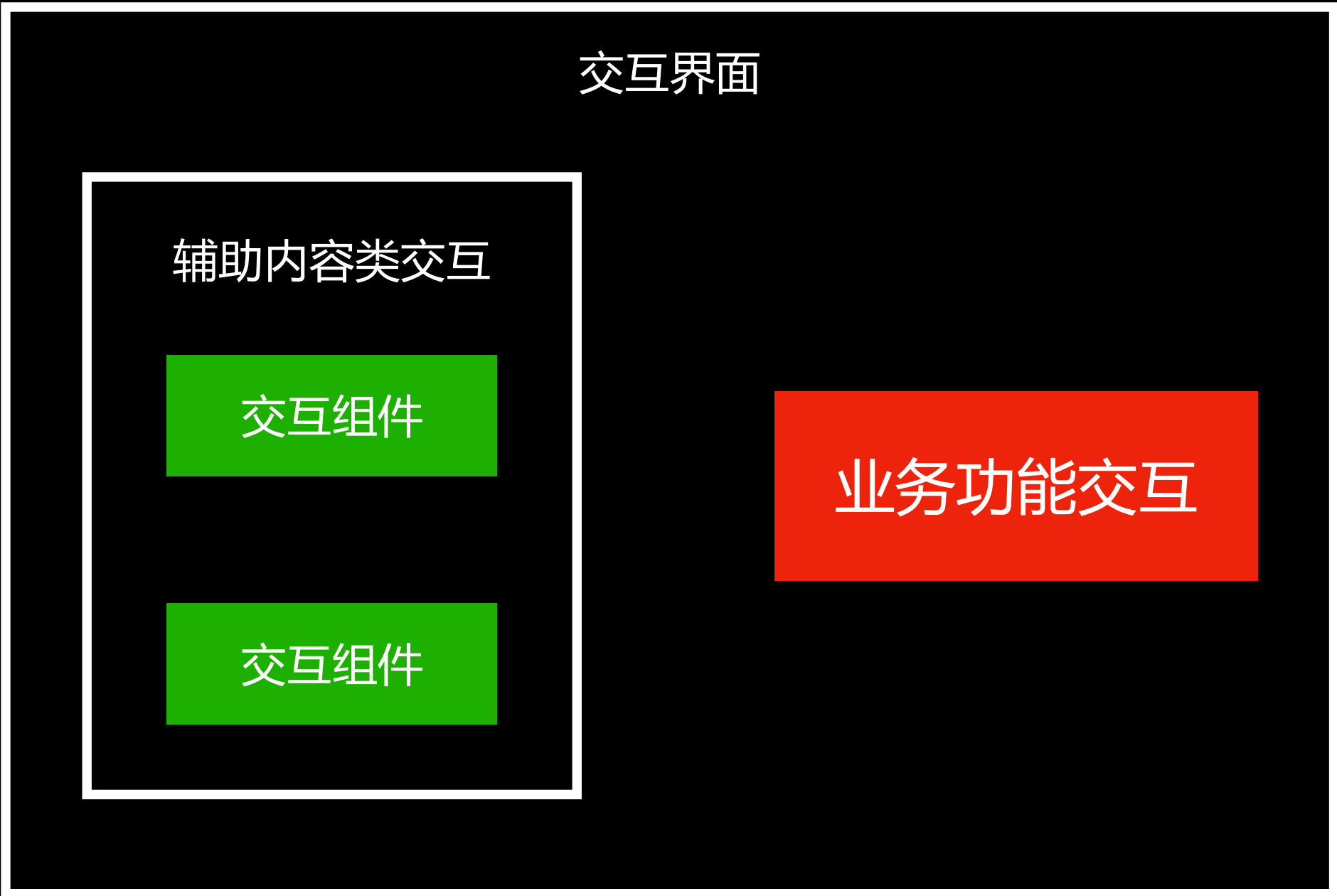
嵌入式视频演示

配置化弹窗/弹层

配置化向导

Step 2

将交互组件按规则动态渲染到界面

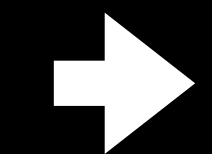


Q: CSS选择器/XPath太专业，用户不会配怎么办？

用户不会配置CSS选择器或XPath，而且配置好的规则可能会因为改版而无效

- 解法
1. 配合技术特征，利用视觉特征相似度做模糊匹配；
 2. 用户只需要选择出视觉特征和偏差范围。

- ☒ 匹配大小
- ☒ 匹配位置
- ☒ 匹配边框/颜色
- ☐ 匹配颜色
- ☒ 匹配结构
- ☐ 匹配文本

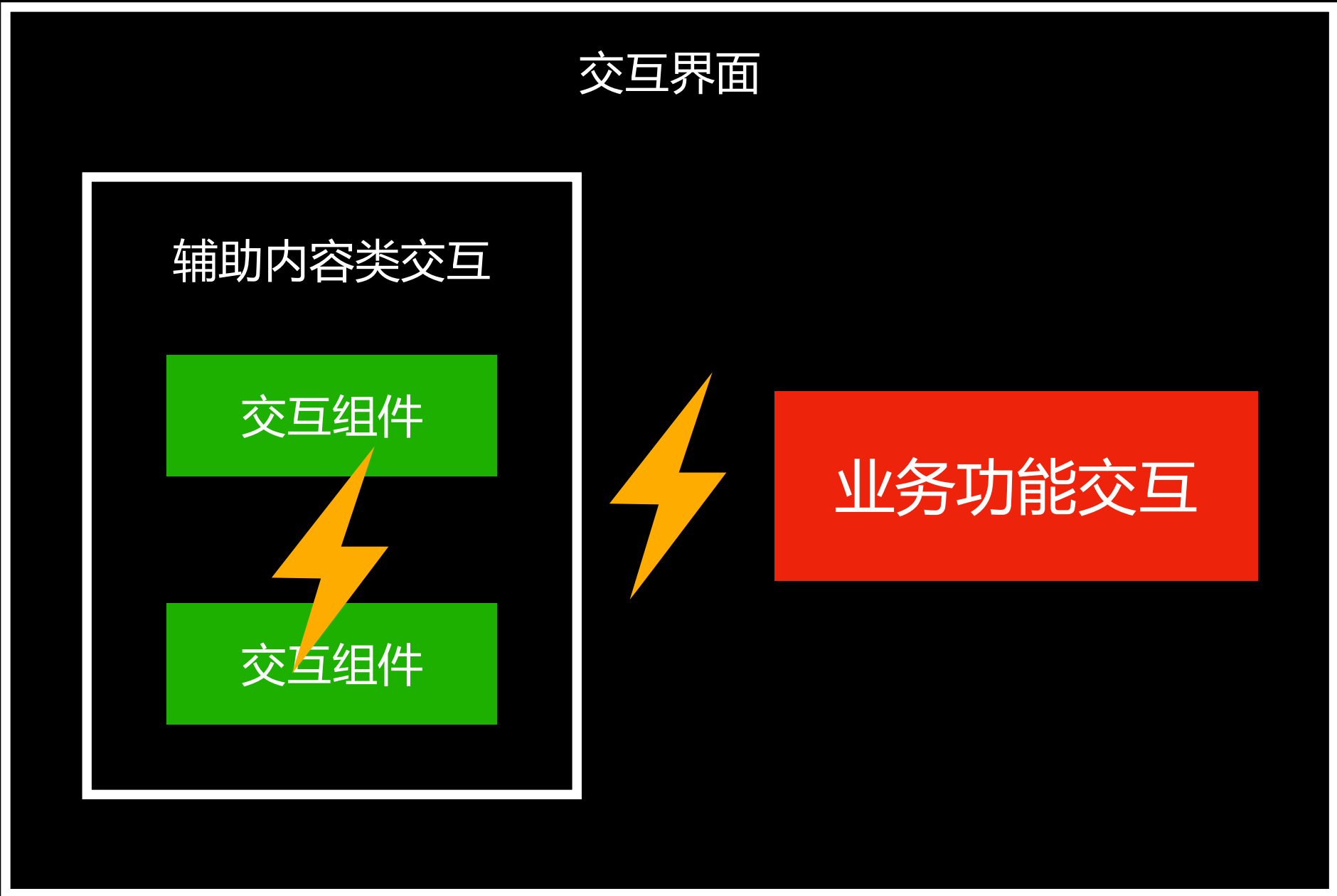


生成

```
1 let $result := (  
2   (: 在/html/body中找所有div或span :)  
3   for $x in /html/body/(div|span)  
4     where  
5       (: 查找宽度在10~60px高度在10~20px范围 :)  
6       (: 查找水平位置在80~136px的 :)  
7       ia:rectMatch($x, "10~60x10~20:80~136") and  
8  
9       (: 且dom深度在5~10的 :)  
10      ia:depthMatch($x, "5~10")  
11   return $x  
12 )  
13  
14 let $result := (  
15   for $x in $result  
16     where  
17       (: 在结果中找边框宽度为1px的 :)  
18       ia:styleMatch($x, "borderWidth:1px")  
19   return $x  
20 )  
21 return $result
```


Step 3

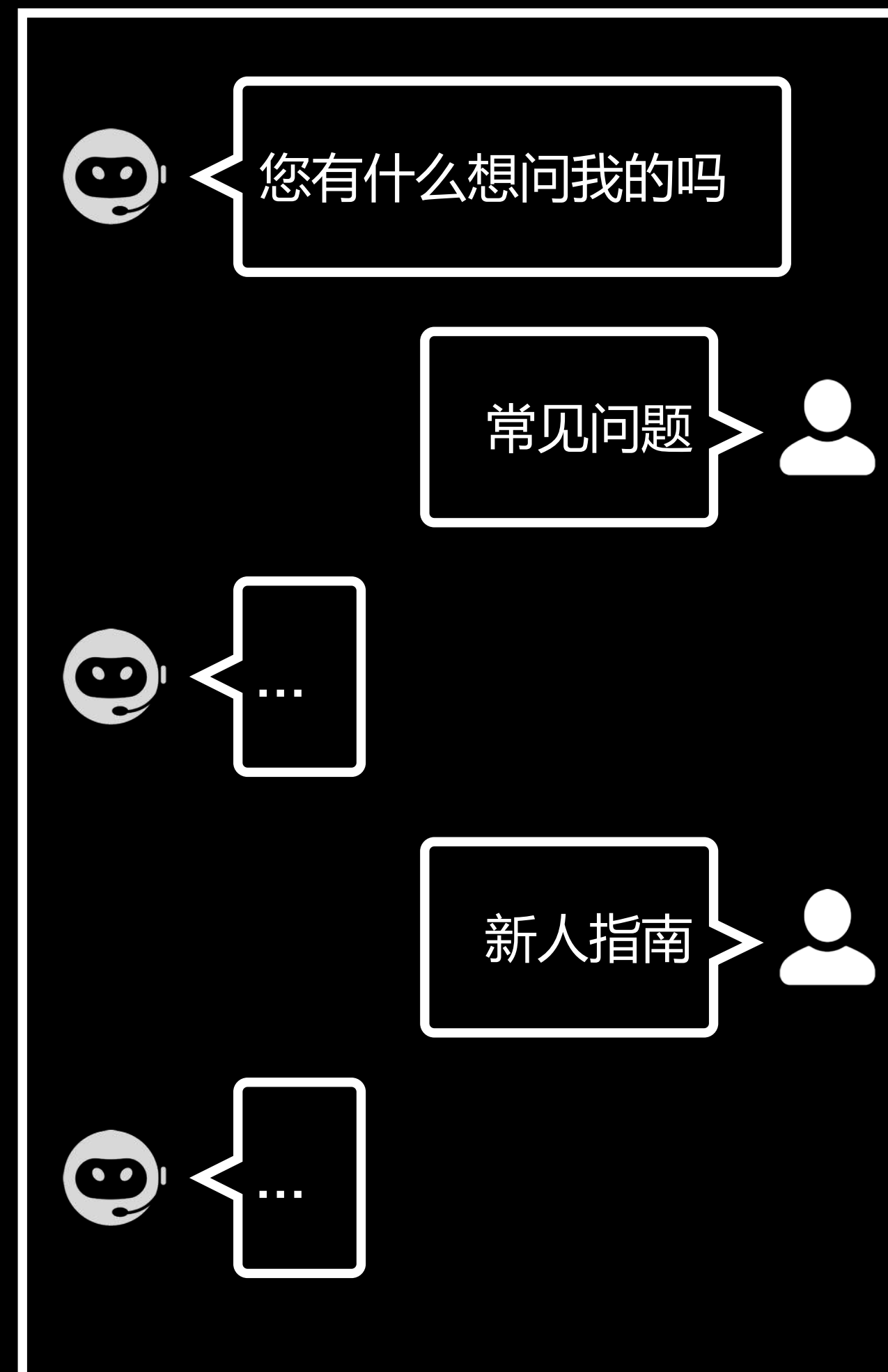
配置交互组件，通过URL触发动作



-  `ia:/ia/frame?url=yuque.antfin.com/...`
内嵌外部页面
-  `ia:/ia/robot/chat?send=套餐怎么配`
向机器人提问
-  `ia:/helpdesk/ticket`
提交工单
-  `ia:/ia/kb/view?id=491`
打开知识库文档
-  `ia:notice?content=😞 **注意啦!**`
显示消息
-  `ia:copy?content=需要复制的文本内容`
复制内容
-  `ia:event?type=click&id=btn1`
转发点击事件，代替用户点击#btn1元素
-  `ia:notice?type=dialog&content=`
打开弹窗，内容为markdown(一幅图片)

Q: 如何用URL完成比较复杂的交互动作？

Step 1. 提示用户即将开始服务
`ia:notice?content=欢迎使用新手引导`



Step 2. **2秒后**，向机器人提问“常见问题”
`ia:robot/send?content=常见问题`

Step 3. **若首次使用**，则紧接着提问“新人指南”
`ia:robot/send?content=新人指南`

Q: 如何用URL完成比较复杂的交互动作？

解法

不可控？

1. 使URL可控，按设定执行

@delay=2s

延迟2秒后url生效

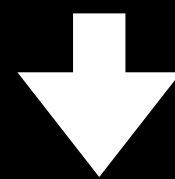
@if=condition

condition成立时url生效

串不起来？

2. 串联URL，让排队执行

ia:url1::url2::url3...



```
<a href="ia:notice?content=欢迎使用新手引导;robot/send?content=常见问题&@delay=2s;robot/send?content=新人指南&@if=!counter.scene">我是新手</a>
```




D2 前端技术论坛
D2 FRONTEND TECHNOLOGY FORUM

精心

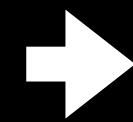
03

策略编排生成逻辑代码

面向复杂交互逻辑的方案

方案核心目标

所有交互逻辑



所有交互逻辑

ProCode开发

前端

低/无代码配置

开发：前端 / 后端 使用：产品技术角色

方案实施效果

一例典型高复杂度表单

3种场景，33个状态，82条逻辑规则

开发进行到5工作日时风险，转用策略编排，用时~2工作日

编译产出代码：~3000行

面向复杂交互逻辑的方案简介



复杂交互逻辑问题

转变为 

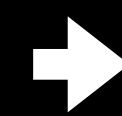
解决 复杂条件 复杂联动 下的 状态管理 问题

决策编排复杂条件

代码示例

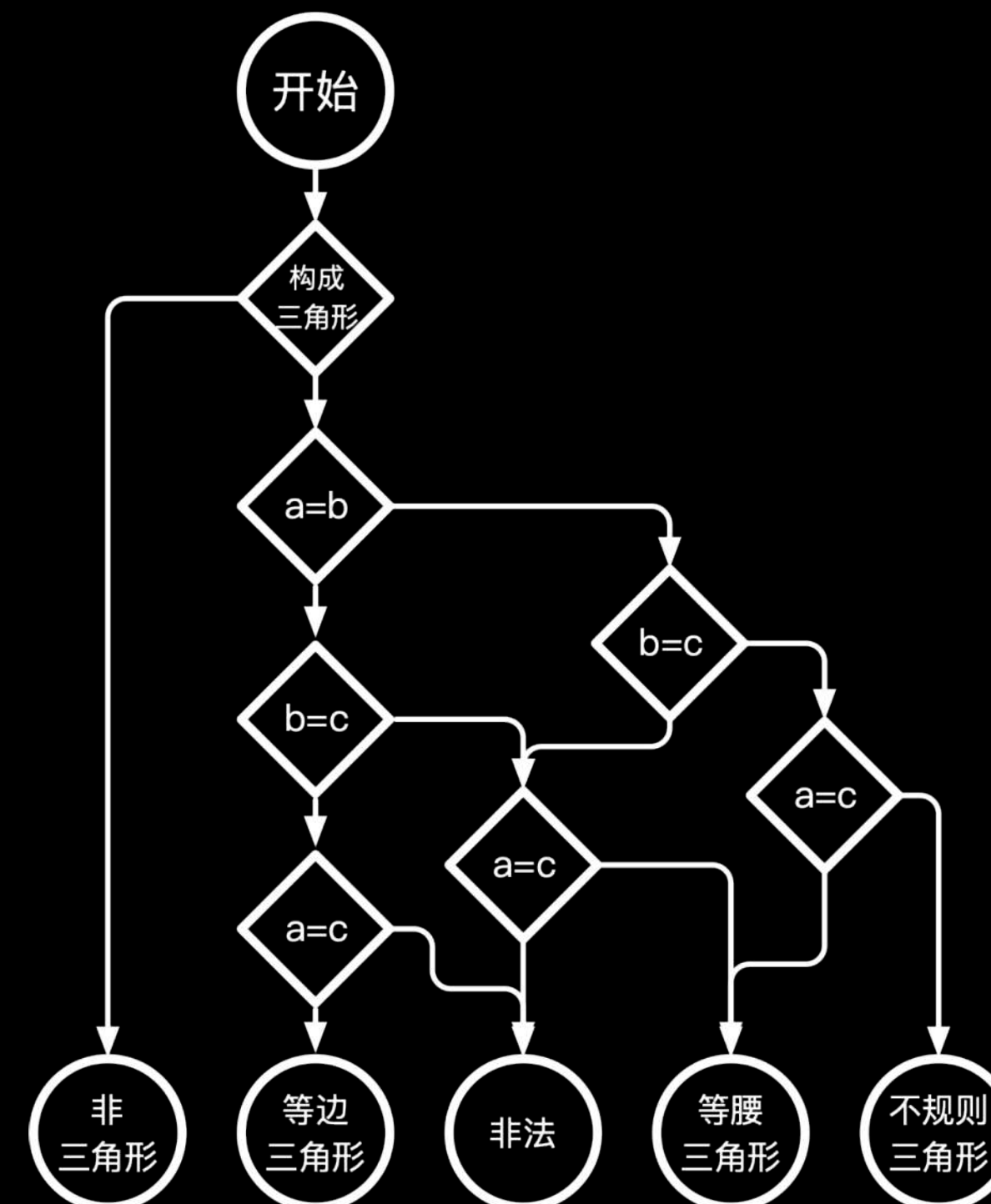
经典if...else嵌套结构

```
if (构成三角形) {  
  if (a=b) {  
    if (b=c) {  
      if (a=c) return 等边三角形  
      else return 非法  
    } else {  
      if (a=c) return 非法  
      else return 等腰三角形  
    }  
  } else {  
    if (b=c) {  
      if (a=c) return 非法  
      else return 等腰三角形  
    } else {  
      if (a=c) return 等腰三角形  
      else return 不规则三角形  
    }  
  }  
} else  
  return 非三角形
```



流程编排方式

类BPM方式，偏重复杂流程表达

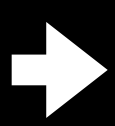


决策编排复杂条件

示例代码

卫述句式化简条件结构

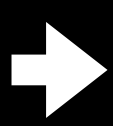
```
if (构成三角形) {  
  if (a=b && b=c && a=c) return 等边三角形  
  if (a=b &&!b=c &&!a=c) return 等腰三角形  
  if (!a=b && b=c &&!a=c) return 等腰三角形  
  if (!a=b &&!b=c && a=c) return 等腰三角形  
  if (!a=b &&!b=c &&!a=c) return 不规则三角形  
}  
if (!构成三角形) return 非三角形  
return 非法
```



决策表

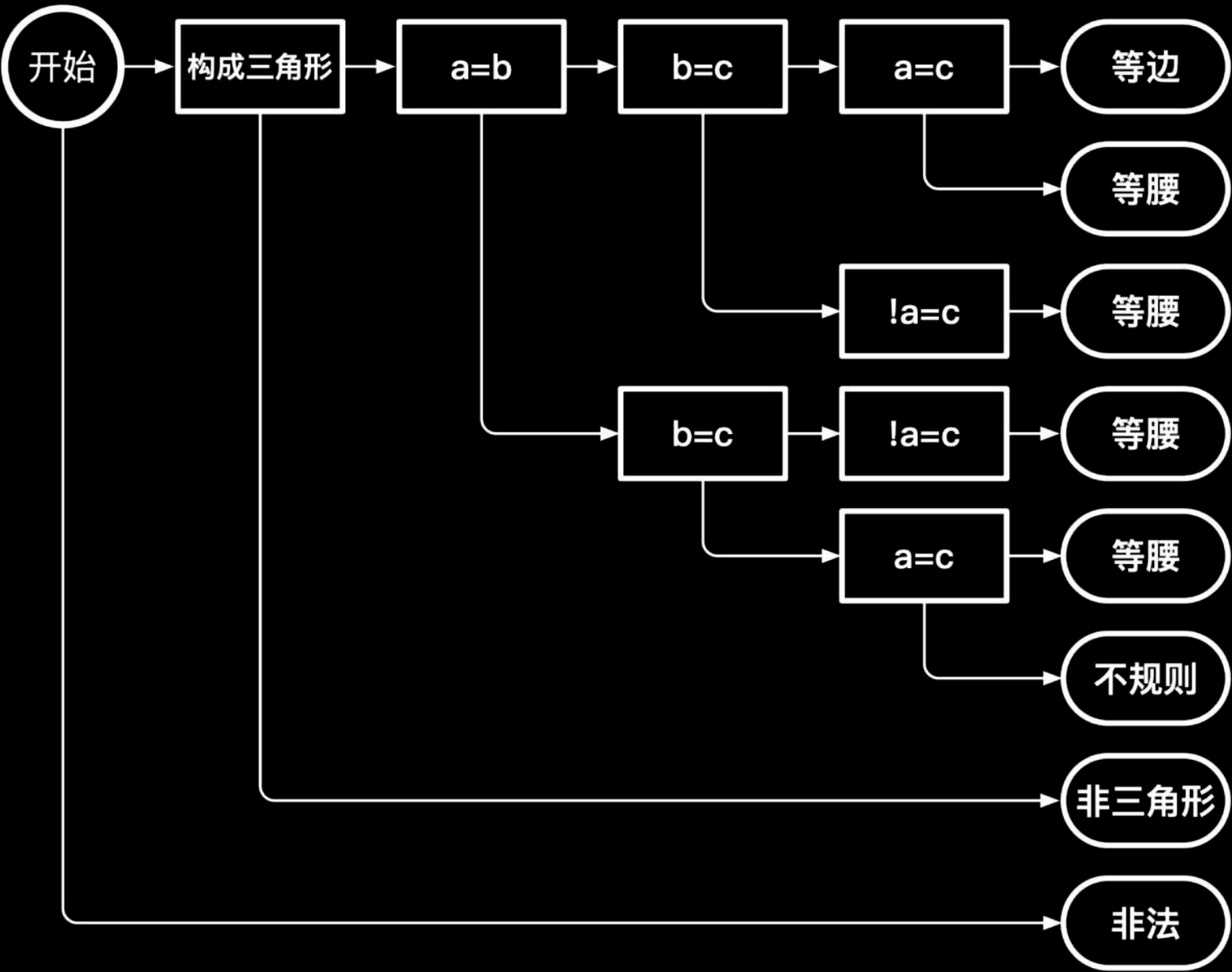
DMN方式，偏重复杂逻辑表达

构成三角形	a=b	b=c	a=c	目标状态
○	○	○	○	等边三角形
○	○	×	×	等腰三角形
○	×	○	×	等腰三角形
○	×	×	○	等腰三角形
○	×	×	×	不规则三角形
×	-	-	-	非三角形
-	-	-	-	非法



决策编排方式

决策表转为决策树



决策编排复杂联动

示例代码

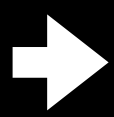
经典联动逻辑

// 当A的值改变时计算C

```
onChange(() => {  
  if (A && !B) {  
    C = 0  
  } else {  
    C = 1  
  }  
}, [A])
```

// 当B的值改变时计算C

```
onChange(() => {  
  if (B && !A) {  
    C = 2  
  } else {  
    C = 3  
  }  
}, [B])
```

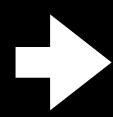


决策表

利用两张决策表表达逻辑

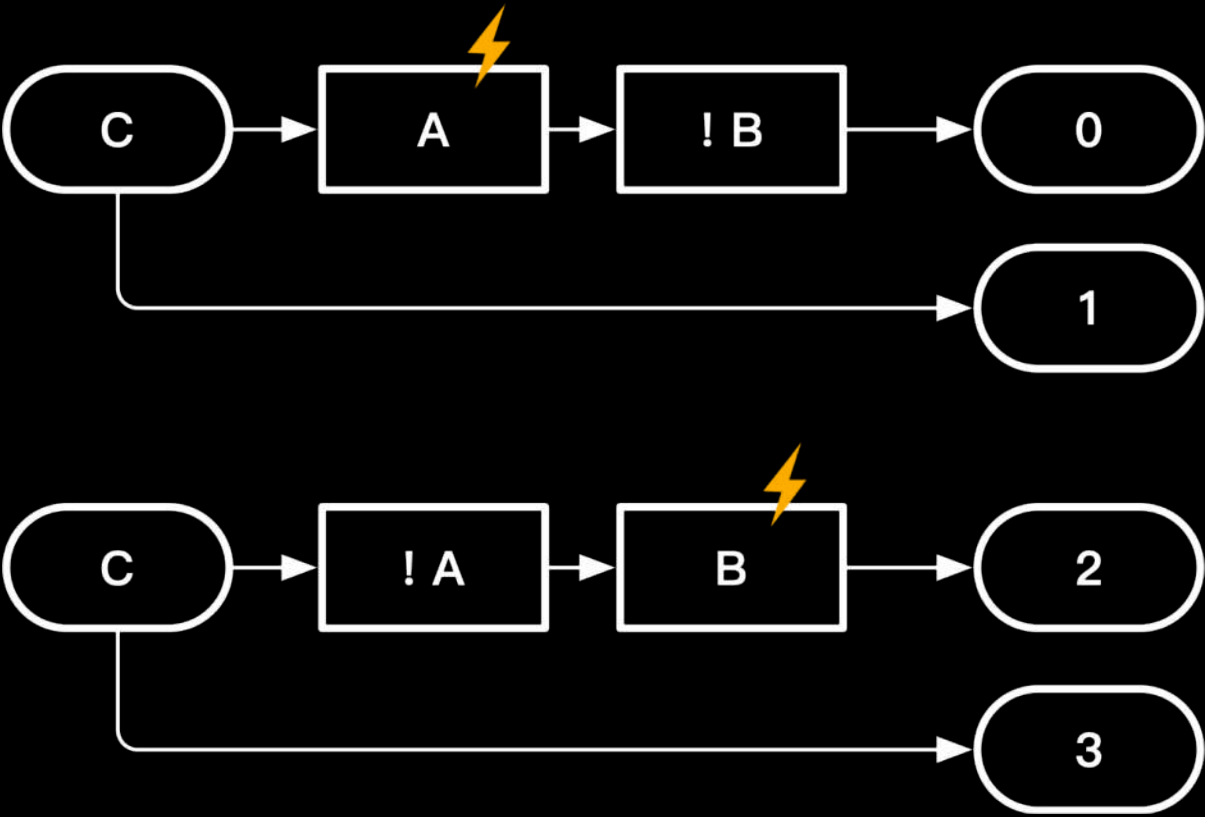
A	B	C
○	×	0
×	-	1
-	○	1

A	B	C
×	○	2
-	×	3
○	-	3



决策编排方式

为状态增加触发事件



决策编排状态分治管理

贷款类型

公积金

商业贷款

还款方式

等额本息

等额本金

自由还款

贷款利率

3.5%

贷款期数

累计利息

待计算

计算

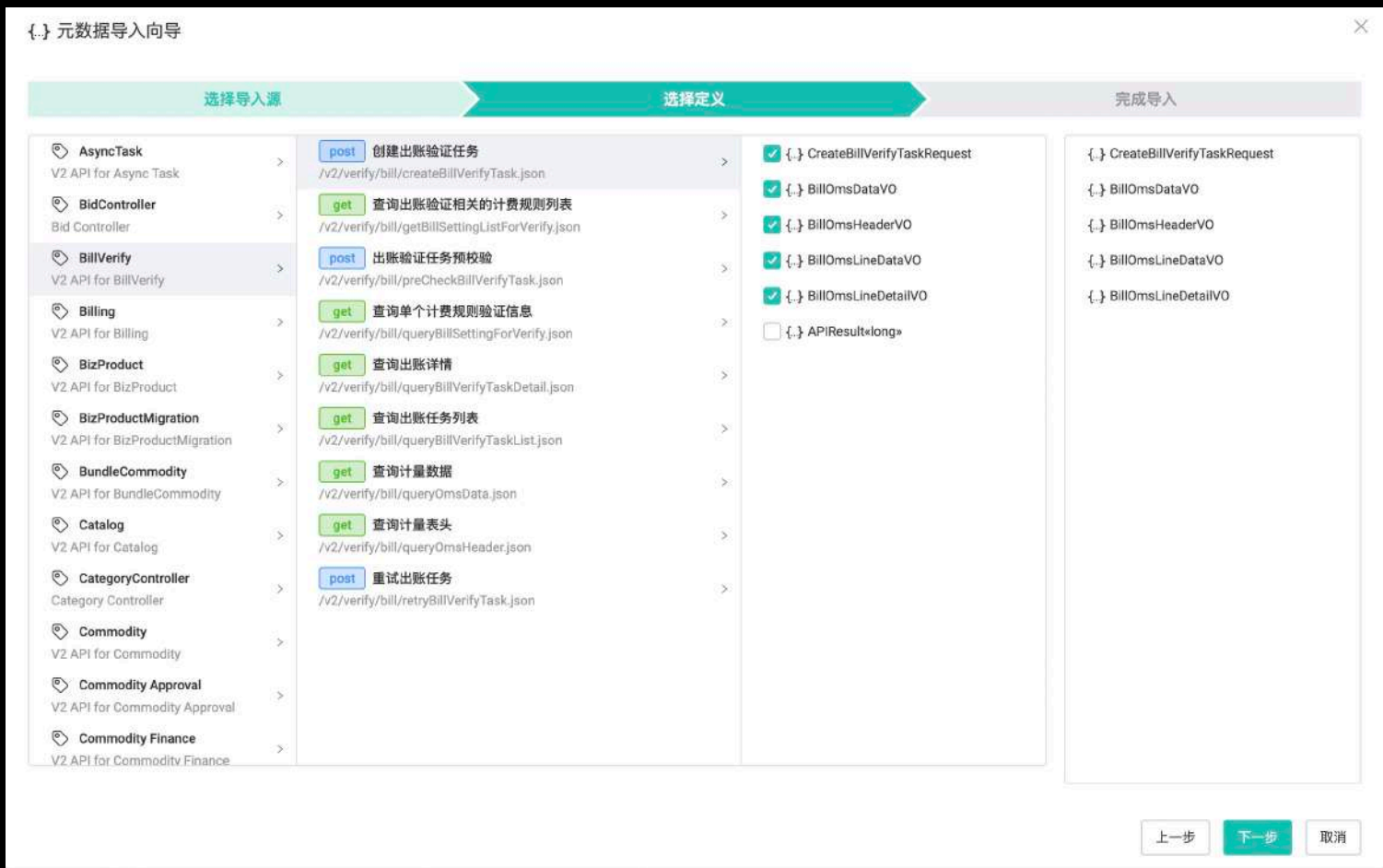


策略编排方案实践

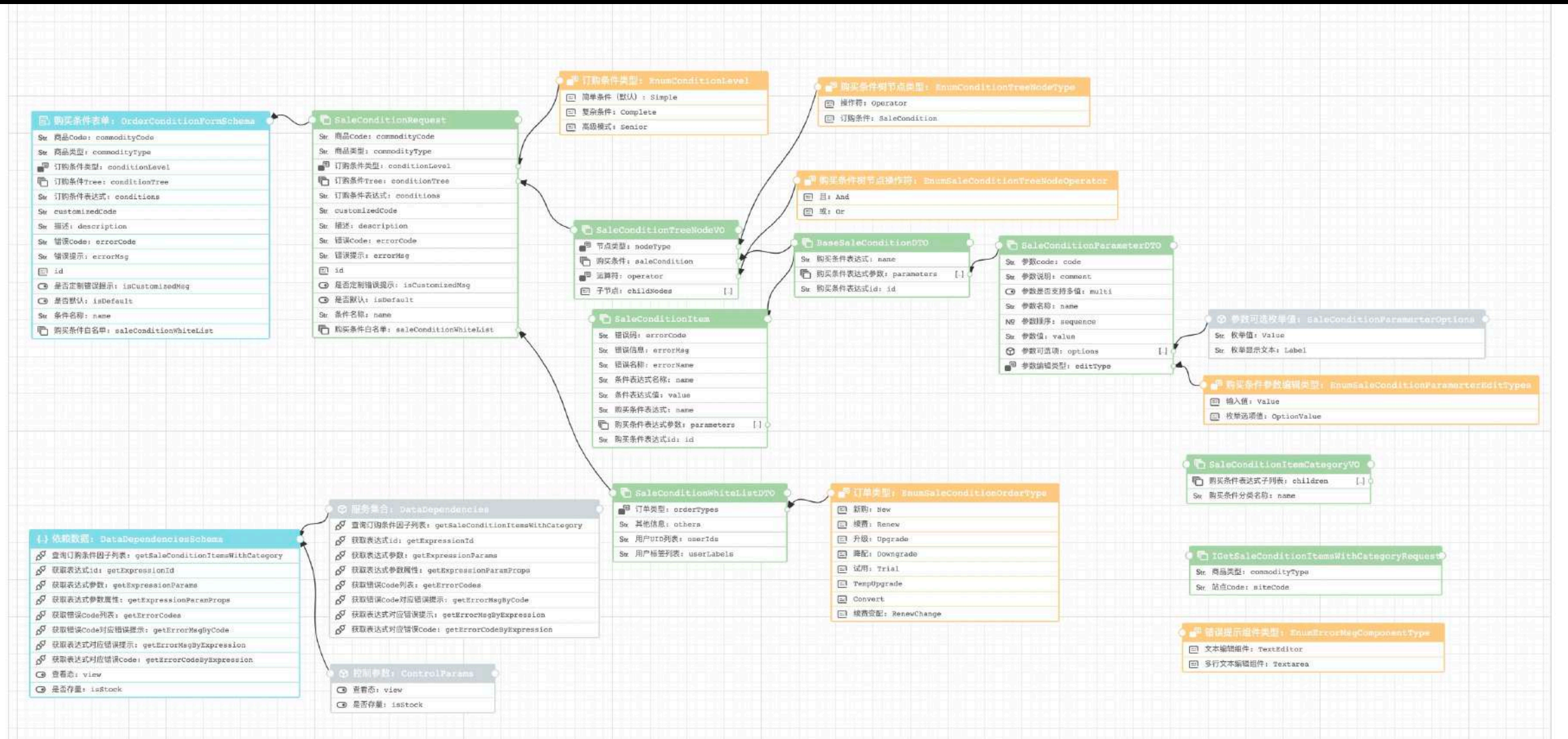
Step 1

定义要策略编排的对象

元数据定义了即将策略编排的对象



导入后端接口定义

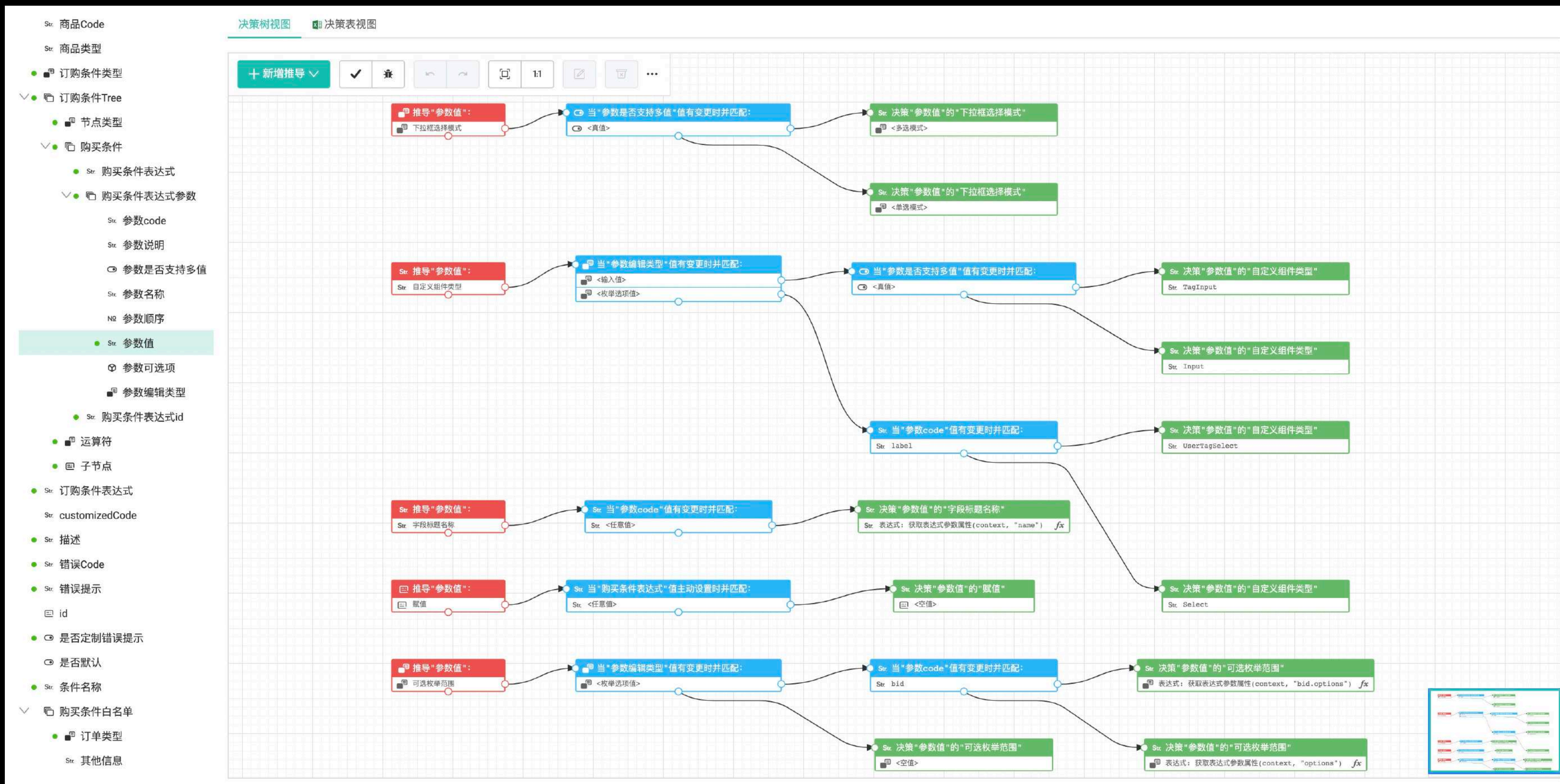


自行定义元数据

Step 2

进行策略编排

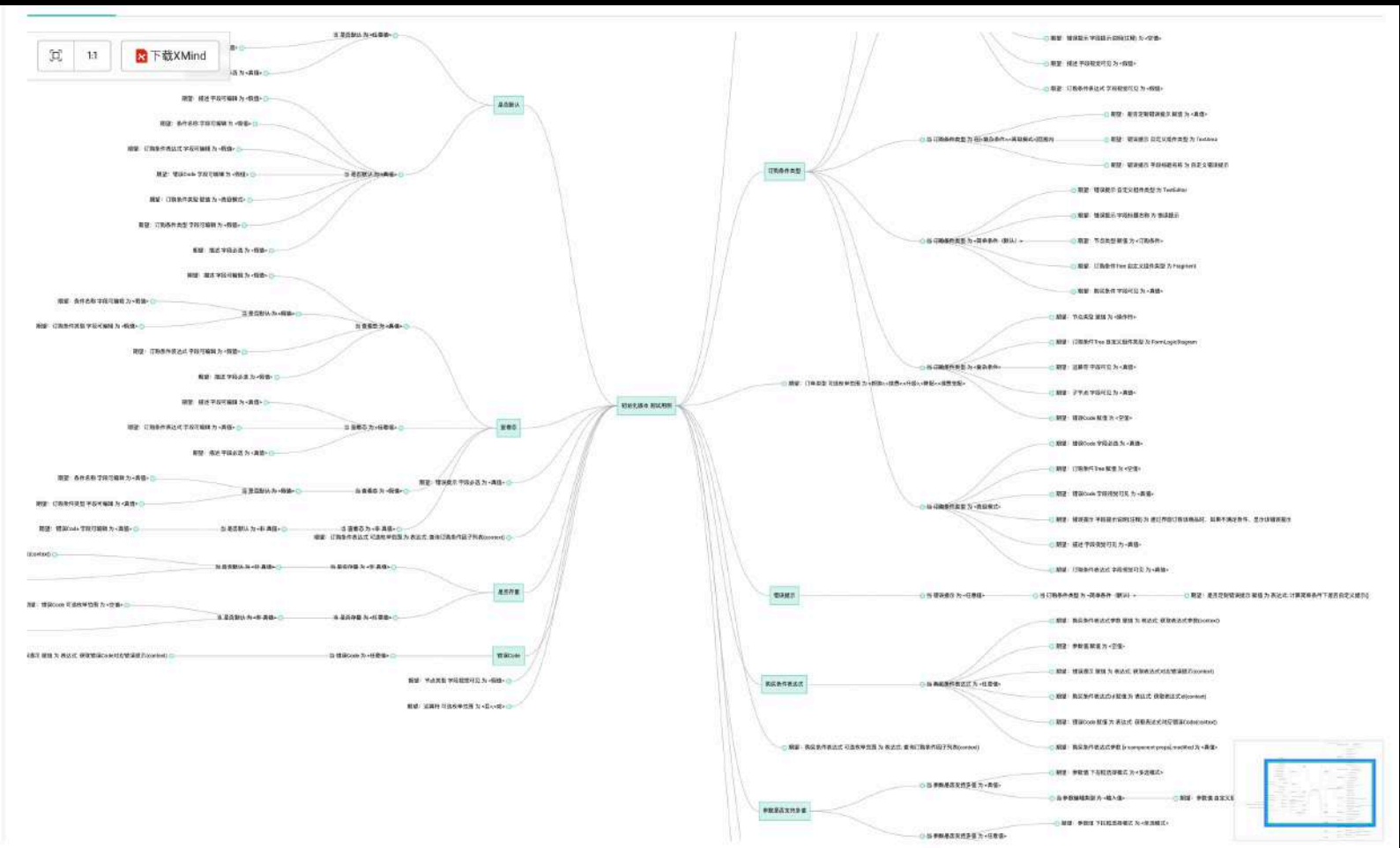
将逻辑关系、联动关系，以策略编排的形式完成



Step 3

调试逻辑及联动

测试驱动开发，保证规则在发布前经过充分验证

元数据

Step 4

将策略编译为专业的源码

输出代码要规范严谨，注释详尽易懂

```
317 /**
318  * 是否定制错误提示(isCustomizedMsg)联动逻辑定义
319  * @param context 联动逻辑依赖的上下文
320  */
321 effects.isCustomizedMsg = async (context: IDataContext) => {
322   const { triggerState, triggerType, values, exp } = context;
323   const states: ISubEffectResult<ISubEffectState<boolean>>[] = [];
324   const fieldName = "isCustomizedMsg";
325   let matches = null;
326
327   // 当初始化时，立即设置字段的默认属性:
328   if (triggerType === EnumTriggerType.Initial && !triggerState) {
329     const state = getEffectState(states, fieldName, matches);
330     return states;
331   }
332
333   // 执行新判断前清空原有匹配
334   matches = null;
335
336   // 当订购条件类型的值主动设置时，立即设置字段的“赋值”：
337
338   // 当错误提示的值有变更时，立即设置字段的“赋值”：
339   if (
340     (triggerState.name === "conditionLevel" &&
341       triggerType === EnumTriggerType.Initiative) ||
342     (triggerState.name === "errorMsg" &&
343       triggerType === EnumTriggerType.Passive)
344   ) {
345     // 首先设置命中检测标记，以防止出现多重命中
346     const checkPoint = createCheckPoint();
347     const state = getEffectState(states, fieldName, matches);
348
349     // 决定字段的“赋值”的条件(当前1/共2):
350     if (
351       // 订购条件类型 为 简单条件 (默认)
352       values.conditionLevel === EnumConditionLevel.Simple &&
353       // 错误提示有变更时
354       IGNORE_CONDITION &&
355       triggerState.name === "errorMsg"
356     ) {
357       // 进行命中检测，如果之前已命中过，则抛出异常
358       checkPoint.check();
359
360       // 设置是否定制错误提示的赋值为：计算简单条件下是否自定义提示()
361       FieldUtils.setValue(
362         state,
363         await fn.isSimpleLevelCustomErrorMsg.bind(context)()
364       );
365     }
366   }
367 }
```

```
/* 定价支付方式 为 预付费 */
values.basicInfo.chargeType === EnumChargeType.PREPAY &&
/* 编辑模式 为 新增 */
values.editMode === EnumEditMode.CREATE &&
/* 商品模板类型 不为 实物类商品模板 */
values.commodityTemplateType !== EnumCommodityTemplateType.HARDWARE
/* 商品状态 为 订购时长新商品 */
values.commodityStatus ===
  EnumCommodityState.ORDER_TIME_FIXED_COMMODITY
) {
  /* 设置目标属性(value)为：按订购时长计算出的价格类型转换逻辑() */
  states.basicInfo.priceTypeTransforms.value = fn.calcPriceTransform(
    context
  )();
}
if (
  /* 定价支付方式 为 预付费 */
  values.basicInfo.chargeType === EnumChargeType.PREPAY &&
  /* 编辑模式 为 新增 */
  values.editMode === EnumEditMode.CREATE &&
  /* 商品模板类型 不为 实物类商品模板 */
  values.commodityTemplateType !== EnumCommodityTemplateType.HARDWARE
  /* 购买方式 为 按周期购买 */
  values.commodityBuyType === EnumCommodityBuyType.BY_CYCLE
) {
  /* 设置目标属性(value)为：按订购时长计算出的价格类型转换逻辑() */
  states.basicInfo.priceTypeTransforms.value = fn.calcPriceTransform(
    context
  )();
}
```

可将源码实时编译，前端直接引入

可将源码提交到git仓库，保证延续性

可将源码构建发布到cdn / npm

可发布为api作为web服务

04

Next

一体化赋能



D2 前端技术论坛
D2 FRONTEND TECHNOLOGY FORUM

精心

thanks