# antflow快速上手手册二之自定义处理逻辑

相信通过快速上手手册一,你已经能够跑起低代码(LF)或者自定义流程(DIY)了,但是很多时候业务并不都是这么简的,很多时候需要根据流程流转状态定制一些动作。比如一个请假流程完成以后根据用户请假的天数扣除他相应的假期天数。（这里只是举一个例子，实践中请假流程可简单可复杂，很多公司请假流程并没有在流程完成以后自动扣除用户假期天数操作，而是人事查看流程核对以后手工在页面扣除用户假期）。

## 流程完成时执行动作。

流程完成时执行一个动作（即插入用户业务代码）是最为常见的需求。低代码（LF）和自定义（DIY）流程处理api大致相似，只是实现的接口不是同一个。

### LF流程处理

编号一个自定义的类，实现ILFFormOperationAdaptor，并加在类上加上LFFormServiceAnnoAttribute 注解，svcName为流程的formCode（用户在前端设计时定义的） ，desc对于LF流程来说没有业务含义，相当于注释。在OnFinishData里面编写相应的业务代码。

### DIY流程处理

和低代码流程不同的是，DIY流程需要实现IFormOperationAdaptor，它是个**泛型接口**，必须是**BusinessDataVo**或者其**子类**，前面讲过，主要用于存储流程运行时信息和前端提交的表单信息。和LF流程另一个不同时其上加的属性是DIYFormServiceAnnoAttribute，svcName即为流程的formCode（LF流程的是在前端页面添加的，DIY的formcode在这里首次添加的），添加完以后流程会扫描所有实现了IFormOperationAdaptor的类，并拿到它们的属性里的svcName，这样就知道DIY流程都有哪些了。（LF流程的是用户在页面添加的，存在了字典表里面）。后面就一样了，在OnFinishData里面写业务代码

## 流程打回时执行动作

通过前面流程完成时执行动作，我们已经添加了一个自定义的类来处理完成时动作。流程打回动作同样也是在这个自定义类里面实现的。Antflow在接口里预先定义了多个流程不同阶段回调函数，方便用户在不同阶段插入自定义处理逻辑。这样通过接口约束形式将流程不同阶段的事件处理都聚集在一个类里面，方便后面快速找到入口，对于维护海量流程是很有用的。

DIY和LF流程需要插入代码的方法都是OnBackToModifyData方法

1. 将流程提交数据持久化

需要特别说明：LF流程表单数数据持久化是由流程引起来做的。不需要用户处理，这里仅只针对DIY流程

前面多次提到过，流程表单数据是存在BusinessDataVo或者继承自它的子类，用户实现的IFormOperationAdaptor接口的泛型参数即是它。

用户在OnSubmitData里面拿到Vo对象里的表单字段数据（字段名和类型都是自己定义的、和前端约定好的），用户提交流程时流程引擎会回调OnSubmitData方法，并且把用户提交的表单数据带过来，用户从中取出需要入库的字段，存入数据库中。用户可以参考第三方账号申请demo流程ThirdPartyAccountApplyService。

## 流程预览时塞入条件字段

同样，本方法不适用于LF流程，LF流程完全由引擎来处理。

之所以要在预览时塞入条件字段，这是因为antflow流程流转和表单是松耦合的，对于DIY流程，引擎对表单字段是完全无感知的，并且引擎也不关心表单业务。但是有些流程有条件分支，条件往往是由表单中一个或者多个字段决定的。用户需要把条件字段摘出来，然后放到BpmnStartConditionsVo实体当中。用户同样可以参考第三方账号申请ThirdPartyAccountApplyService，第三方账号申请账号的类型是条件，会影响流程走向。

方法是PreviewSetCondition，用户在里面插入代码

如果用户的DIY流程没有条件，这个方法返回一个new BpmnStartConditionsVo即可。

## 流程启动时塞入条件字段

同样，本方法不适用于LF流程，LF流程完全由引擎来处理。

原因和上面流程预览时塞入条件字段相同，流程引擎并不知道哪个字段是条件字段，需要用户把它抽出来放到BpmnStartConditionsVo里面（一般和BusinessDataVo里保持一致，但是不是必须）。基本上拷贝上面PreviewSetCondition里面的内容，然后放到LaunchParameters里面。

## 为流程查看页面（我的发起，我的已办）表单塞入值

同样，本方法不适用于LF流程，LF流程完全由引擎来处理。

在前面OnSubmitData方法里我们将用户表单数据持久化，后面环节需要使用表单数据时，我们就可以从表里面查出来。为用户查看页面提供数据就是一种场景，用户在OnQueryData方法里面查询出持久化的数据，如何查询呢？流程引擎回调方法时会携带一个对象实体，前面说过，它是BusinessDataVo或其子类，里面有个BusinessId字段，即为业务表的主键，用户可以根据它来查询自己的业务表拿到持久化的数据。

用户可能会好奇，前面说antflow引擎和表单是分离的，弱耦合的。那么业务Id是如何拿到的呢？这里注意是弱耦合。在bpm\_business\_process表里面存储了流程和表单的关联关系。流程发起以后，就会生成流程表编号，有了流程编号就能根据流程编号查在bpm\_business\_process里查到一条记录，记录里面存了业务数据表主键。（存储的时候是如何知道哪个字段是主键呢？其实是一种约定，在OnSubmitData里，用户将数据持久化以后，需要将主键赋值到BusinessId字段上（用户可以任意赋值，但是随便赋值就会导致后面查不到数据，业务不正确）。赋值完以后引擎就能拿到业务主键了，在适当的时机将流程编号、流程实例id和业务id存到bpm\_business\_process表里面。

**bpm\_business\_process表是antflow里面最核心的一张表，用户一定要记住它！**