工作流

平台提供了自己的工作流引擎，基于该工作流，使用者能方便地在界面上进行流程的配置及绘制、任务执行人员的分配、流程运行的监控等，工作流部分也提供了一系列接口，让使用者能方便进行扩展。

1. 基本配置及使用
   1. 与流程相关的服务
      1. 流程管理服务

流程管理服务是流程配置的入口，我们配置的所有流程都通过这个服务进行管理。

流程管理服务编码为“SY\_WFE\_PROC\_DEF”，使用了自定义的卡片页面procDef.jsp。



“方法&按钮”页面中定义了一系列的操作。用于对流程定义进行维护。



* + 1. 流程公共按钮服务

服务SY\_WFE\_PROC\_DEF\_ACT中定义了了一系列的操作，作用在于：在流程运行中，提供与流程运行相关的功能，如流程跟踪、锁定等。这些操作包括：



操作的名称必须是以cm开头。

如果想添加新的操作，在SY\_WFE\_PROC\_DEF\_ACT的“方法&按钮”中添加即可。



* 1. 一个简单流程的配置及运行

创建流程的菜单入口如图：



要完整地创建一个流程，需要配置流程基本信息、绘制流程图。

下面，我们从配置一个简单的流程开始。

* + 1. 业务表单必备条件

流程不能孤立存在，它必须有业务表单的数据做支撑。

对业务表单的要求：

1、在设计上必须要有几个必填字段：

|  |  |  |  |
| --- | --- | --- | --- |
| 编码 | 名称 | 字段类型 | 长度 |
| S\_WF\_INST | 流程实例编码 | 字符串 | 40 |
| S\_WF\_STATE | 当前流程状态 | 数字 | 4 |
| S\_WF\_USER | 当前处理用户 | 字符串 | 2000 |
| S\_WF\_NODE | 当前处理节点 | 字符串 | 2000 |

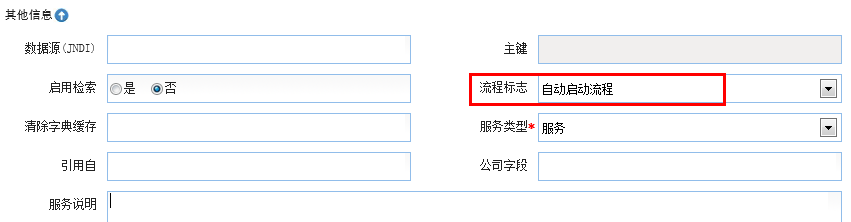
2、在服务配置上，必须配置标题格式、编码格式，这些字段主要用于在待办中与其他服务的待办进行区分：



在待办列表中会这样显示：

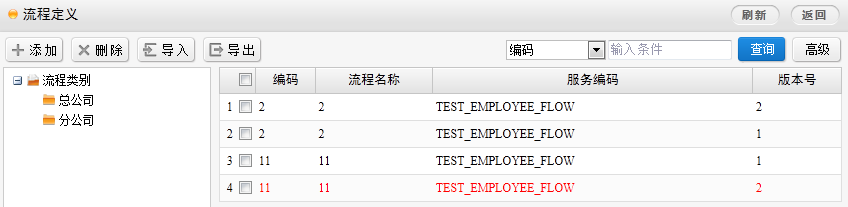


3、如果想在添加数据时自动启动流程，必须将流程启动方式设置为自动启动，默认为手动启动，手动启动意味是需要自己写后台代码启动流程。

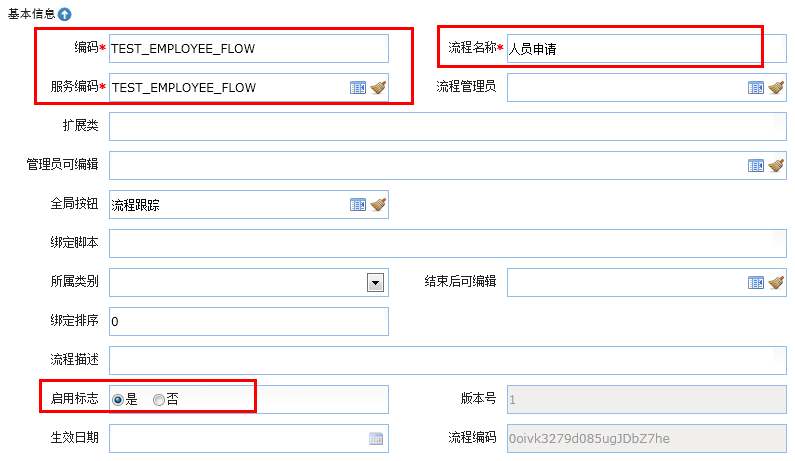


* + 1. 配置流程

进入流程管理页面，可以看到流程按照“流程类别”进行了导航：



点击“添加”进入编辑页面。



在配置流程的基本信息中，先只需关注“编码”、“流程名称”、“服务编码”、“提醒标题”4个必填项。

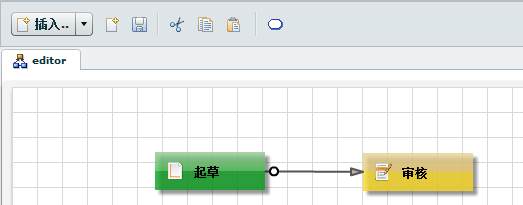
“编码”：流程的唯一标识，为了便于区分，通常会设置所选为“服务编码”的值；

“流程名称”：为流程起一个贴切的名字；

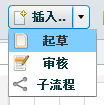
“服务编码”：流程不能孤立存在，它必须有业务表单的数据做支撑，如请假流程的请假单，发文流程中的公文，合同审批流程的合同。

* + 1. 绘制流程图

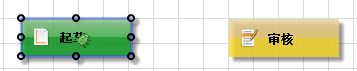
流程基本信息配置好后，开始进行流程的绘制，为了简单，这里只绘制两个节点。



每个流程有且仅有一个“起草”节点，点击“插入”—>“起草”，将鼠标移到工作区单击即可。接着，在工作区绘制一个“审核”节点。



接下来，要绘制两个节点间的连线，在“起草”节点上单击鼠标左键，将鼠标移到节点文字上，此时，出现节点上出现绿色小甲虫图标，单击鼠标左键，将鼠标向“审核”节点移动，在“审核”节点上单击，连线添加成功。

流程的运行少不了人的参与，接下来，我们为两个节点配置处理人。

“起草”节点不用配置处理人，因为流程是在创建表单数据的时候启动的，创建表单的人就是起草人。

“审核”节点的处理人，设定为本处室的部门领导，如图。

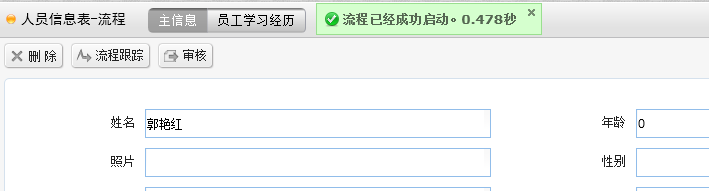


不管是流程基本信息的更改，还是流程图的更改，一定记得点击“保存流程”进行保存。

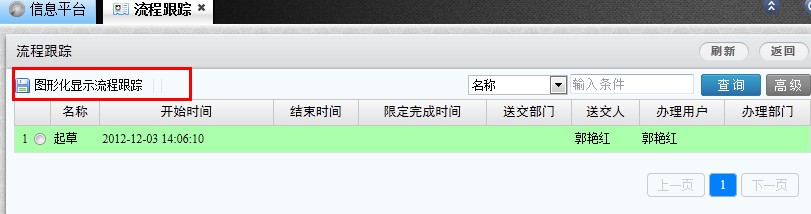
* + 1. 启动流程

流程配置成功后，通过添加流程关联的业务表单数据，可以看到流程运行情况。在本例中，需要添加一条人员信息。

“保存”后，出现下面界面。



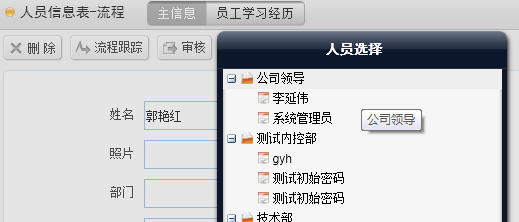
点击“流程跟踪”，显示当前正处于“起草”环节。



点击“图形化显示流程跟踪”，可以图形化显示流程的整个运行情况。



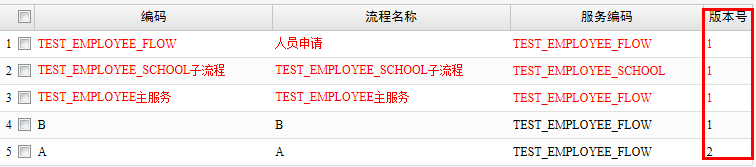
点击“审核”按钮，让流程继续向下运行。



* 1. 流程定义详解
     1. 多版本

为什么需要多版本？

主要场景：原来的流程运行一段时间后，因业务需求发生变化，业务流程也有了调整。此时，旧数据（主要为已经启动了流程的）需要在老流程上运行；而新的业务数据需要走新流程，因此我们需要对流程进行版本管理。如图：



有了多版本后，在创建数据后，会启动哪个流程呢？见*1.5、流程启动*。

除了能自动运行最新版本的流程外，我们可以通过代码的方式运行指定版本的流程，见流程的二次开发。

* + 1. 高级配置

流程基本信息的配置除了前面讲到的必填字段外，主要有以下内容：



|  |  |  |
| --- | --- | --- |
| **配置项** | **使用说明** | **备注** |
| 流程管理员 | 配置流程管理员后，  在流程运行中，流程管理员可修改在“管理员可编辑”中配置的字段；  在流程结束后，可修改在“结束后可编辑”中配置的字段 | 该功能暂不可用 |
| 管理员可编辑 |
| 结束后可编辑 |
| 扩展类 | 对在流程运行界面出现的按钮进行定制，需要实现WfFilter接口 | 详见二次开发 |
| 全局按钮 | 配置流程运行各个环节都会出现的按钮，使用的数据来自“流程公共按钮”服务中的数据 |  |
| 所属类别 | 对流程进行分类管理，使用的是数据字典SY\_WFE\_CATALOG |  |
| 绑定脚本 | 启动流程时，需要根据“绑定脚本”中的条件判断能否启动该流程 | 语法为javascript语法。  如'#USER#' == '@UER\_CODE@' |
| 绑定排序 | 某服务配置有多个流程时，在都满足启动条件时，会启动绑定排序小的流程定义 |  |
| 启用标志 | 设置为true时，才可能被使用 |  |
| 生效日期 | 该版本的流程的生效日期 |  |
| 版本号 | 同一名称的流程，可以有不同版本，版本号只读递增，在同时满足启动条件时，优先使用高版本的流程定义 |  |

* 1. 流程图详解
     1. 节点的分类

系统中的流程节点，从图标上看有3种：

起草节点（开始节点）、普通任务节点、子流程节点。

流程图有且仅有一个起草节点（开始节点），任何节点都可以配置为办结节点。节点细分的话，有以下类型：

|  |  |
| --- | --- |
| 节点类型 | 说明 |
| 起草节点 | 有且仅有一个起草节点 |
| 并发节点 | 此类节点可以创建多个任务，进行处理 |
| 合并节点 | 将之前由于并发产生的多个任务，合并为一个任务 |
| 抢占式节点 | 创建一个任务，分配给多个人，仅需一个人办理 |
| 子流程节点 | 将主子流程关联在一起 |
| 办结节点 | 办结流程，可以有多个 |

* + 1. 节点的配置

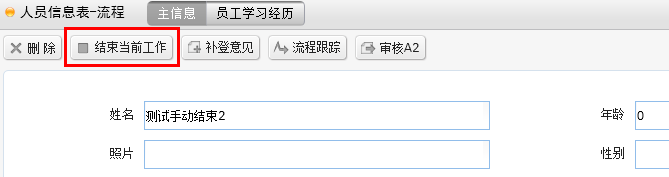
节点配置内容比较多，共有8方面的内容：

* + - 1. 基本信息



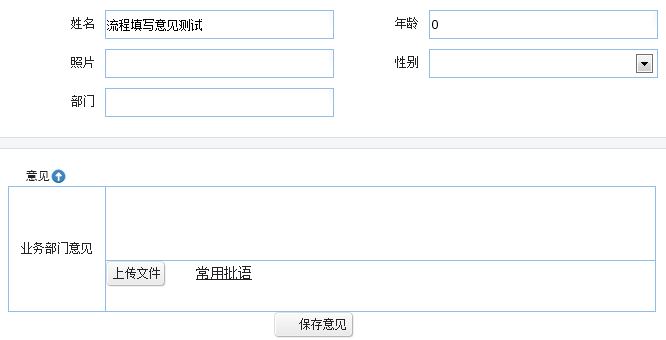
|  |  |  |
| --- | --- | --- |
| **配置项** | **使用说明** | **备注** |
| 自动结束 | 流程任务送交给下一节点后，当前节点的任务是否自动结束，默认为true。  设置为false的场景：用户需要对该任务进行多次处理，然后手动结束该任务 | 见下图 |
| 合并节点 | 通常，合并节点的前面会有分支节点，分支节点在连线上设置。  分支节点会产生多个任务，异步运行。运行到合并节点时，这些任务会合并成一个任务，继续向下运行 |  |
| 意见信息 | 通常在审批、批示环节，需要让处理人填写办理意见，可根据需要设置为必填。 | 见下图 |
| 结束流程 | 设置流程在此节点是否可以办结，用于配置结束节点 | 办结节点可以有多个 |
| 结束按钮名 | “结束流程”设置为true时，出现办结按钮，按钮名字默认为“办结” |  |

“自动结束”设置为false时，需要手动结束当前任务：



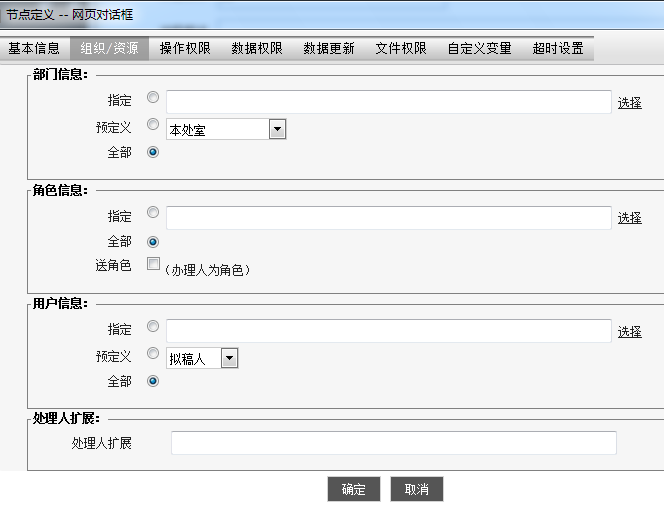
配置审核时必须填写意见：



运行界面出现的意见框： 

* + - 1. 组织/资源

首先注意：“起草”节点不需要设置“组织/资源”中的内容。



该tab页用来指定任务处理人。可以通过部门、角色、用户对处理人进行过滤。多个过滤条件之间是“与”关系。

在部门、用户选择时，有一项叫“预定义”，为了便于理解，这里约定，用户A提交一项任务给用户B时，A叫做sendUser，B叫做toUser。各项“预定义”含义如下：

|  |  |  |
| --- | --- | --- |
| **类别** | **配置项** | **含义** |
| 部门预定义 | 本处室 | sendUser所在的dept及其子部门 |
| 本部门内所有处室 | sendUser所在的tdept及其子部门 |
| 上级机构 | sendUser所在的odept的上级单位内的所有部门；没有上级单位时，取的是SendUser所在的odept内的所有部门 |
| 拟稿机构 | 拟稿人所在的odept及其子部门 |
| 拟稿部门 | 拟稿人所在的tdept及其子部门 |
| 拟稿处室 | 拟稿人所在的dept及其子部门 |
| 用户预定义 | 拟稿人 | 起草节点的处理人，流程发起人 |
| 当前用户 | sendUser |

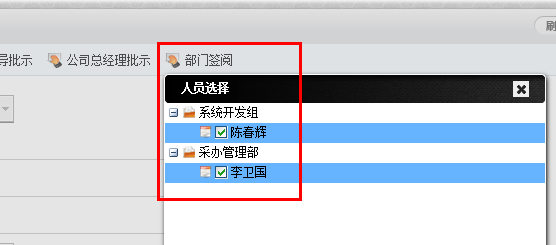
注意：部门过滤选“全部”时，取的是用户所在机构下的所有部门。

配置处理人后，流程将要运行到该节点时，会弹出选择框，让用户更精确地选择处理人。

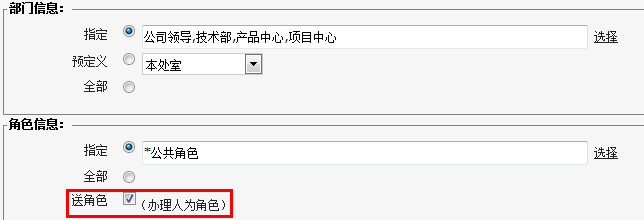
非并发节点，只能选择一个处理人：



并发节点可以多选：



抢占式节点，需要配置“办理人为角色”，或者通过二次扩展



配置出来的抢占式，只能选择到部门，部门内指定角色的人之间进行抢占式办理：



抢占式送交后，通过流程跟踪看，送交给了多个人：



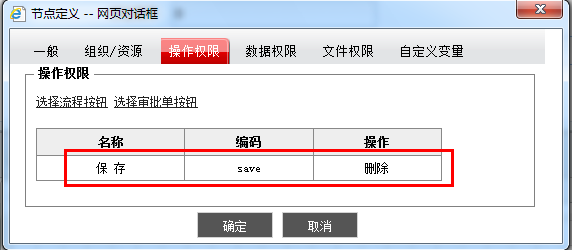
其中一个进行办理：



之后，这个任务就变成该用户的个人任务了。

不能配置出来的，可以使用“处理人扩展”。在二次开发中会进行详细讲解。

* + - 1. 操作权限

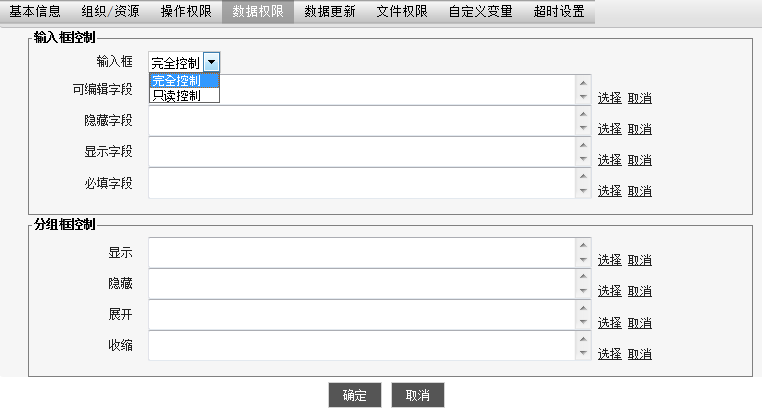


此页面用于配置该任务处理环节支持的操作，具体讲，就是流程运行到该环节，页面上出现的按钮。

流程按钮，“流程公用按钮”服务中定义的操作。

审批单按钮，流程关联的服务的“方法&按钮”，如保存、删除等。

* + - 1. 数据权限



用于对表单的显示进行控制。

完全控制：表单数据是可编辑状态；

只读控制：表单为只读，只有配置在“可编辑字段”中的字段可以编辑。

* + - 1. 数据更新



流程运行时，在3个时机可以设置对数据进行更新。

配置方法如下：

1、更新的值为系统变量或字段运算的结果，“更新内容”不用引号括起来；



2、更新的值为常量，如数字或字符串值，则“更新内容”必须用引号括起来。



* + - 1. 文件权限



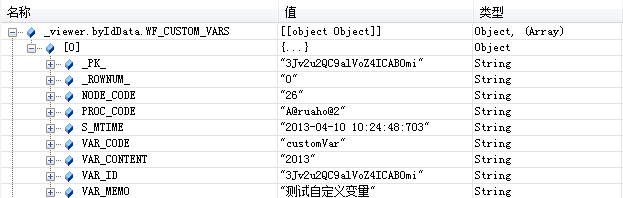
用于控制表单中的正文、附件在各个任务界面的显示与否。

注意：在起草节点，正文、附件都应配置最大权限。在其他任务处理界面，通常只用查看。

* + - 1. 自定义变量



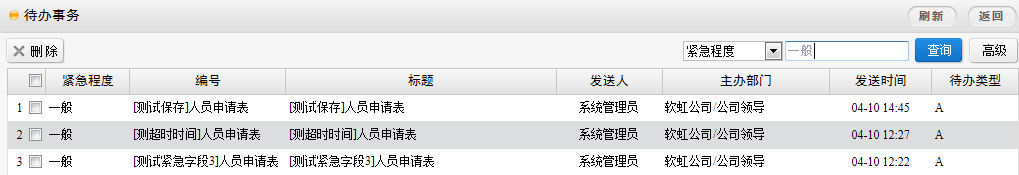
当需要一些特殊的值，而这些值又没法从系统变量、业务表单中读取时，可以自定义变量，在前台可以这样获取：**\_viewer.byIdData[WF\_CUSTOM\_VARS]**

**

* + - 1. 超时设置



不进行设置的话，流程中的任务的紧急程度都是“一般”，在待办中的显示为：



要配置紧急任务的话，业务表单中必须有S\_EMERGENCY字段。它的值会赋给待办服务SY\_COMM\_TODO的S\_EMERGENCY字段。



//TODO 此功能尚有问题

* + 1. 连线的配置

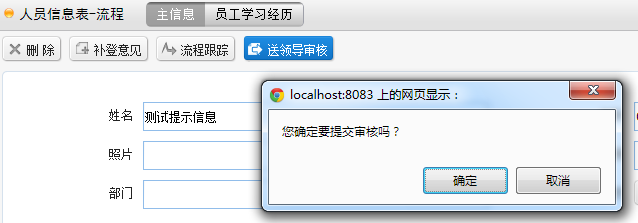
连线代表了流程的流向，在任务处理界面以按钮的形式展示。连线的配置主要有4方面内容。

* + - 1. 基本信息

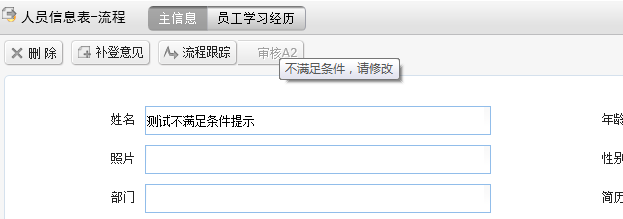


|  |  |  |
| --- | --- | --- |
| **配置项** | **使用说明** | **备注** |
| 允许退回 | 表明流程能否从后节点返回到前节点。  设置为true时，节点上会出现按钮“返回XXX”（XXX为前节点处理人），如果配置了“返回名称”，则退回按钮的名称为返回名称中设置的值 |  |
| 连线排序 | 当一个节点有多条连线时，设置连线排序控制按钮出现的顺序 |  |
| 条件名称 | 显示在连线上的名称，仅为了美化流程图的展示，没有实质意义 |  |
| 送交名称 | 前节点流转到后节点时，出现的按钮的名称 |  |
| 返回名称 | 后节点退回到前节点时，出现的按钮的名称 |  |
| 是否并发 | 设置下一节点为并发节点，并发节点可以生成多个任务，互不干扰地运行 |  |
| 是否出部门 |  | 暂时无用 |
| 扩展类 |  | 暂时无用 |
| 提示信息 | 流转前的提示确认 | 见下图 |
| 未满足条件提示 | 不满足流转条件时的提示。使用时必须配置条件表达式。  注意：不设置此项时，不满足条件的按钮不会显示在界面上 | 见下图 |

流转前的提示信息：



不满足流转条件的提示：



* + - 1. 条件表达式

满足条件表达式中的条件后，流程才能往这个方向流转。

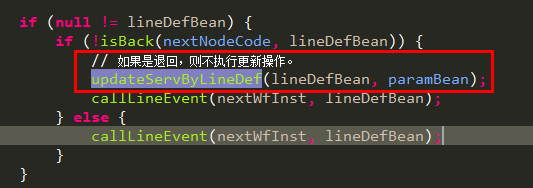


* + - 1. 数据更新

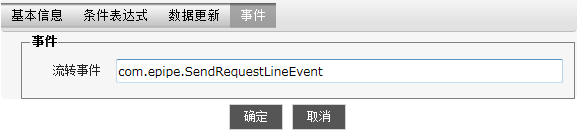
跟节点的数据更新配置是一样的。

注意：连线上的数据更新，只有向前流转时才会执行，退回时不执行。

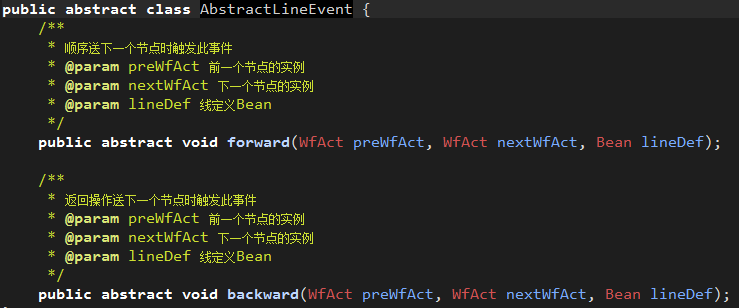
实现代码如下：

****

* + - 1. 事件

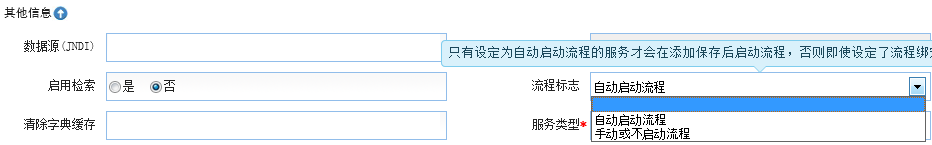


流转事件扩展类需要继承AbstractLineEvent类，流转时会执行。



* 1. 流程启动

流程的启动方式有两种：自动启动，手动启动。在业务表单服务中进行配置的，配置的字段是SERV\_WF\_FLAG:



**自动启动流程**：

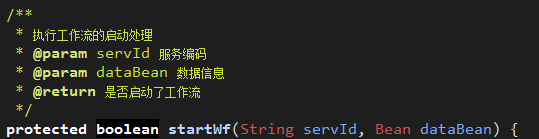
当新建一个业务表单的数据时，相应的流程也会被启动。我们可以在前台界面中通过新建一条数据启动流程，也可以在后台通过代码的形式新建数据启动流程。

**手动启动**：

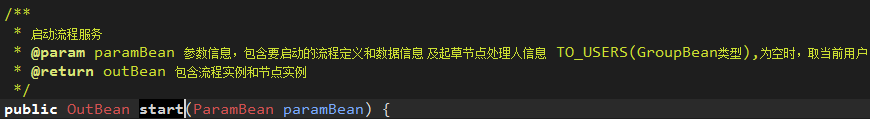
新建数据的时候，不会启动流程。需要使用者通过代码进行调用。

* + - 1. 代码方式启动流程

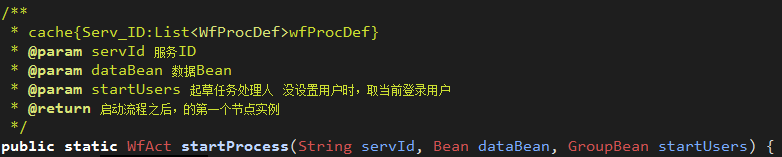
1. 如果类继承了AbstractServ或CommonServ类，可直接使用startWf方法

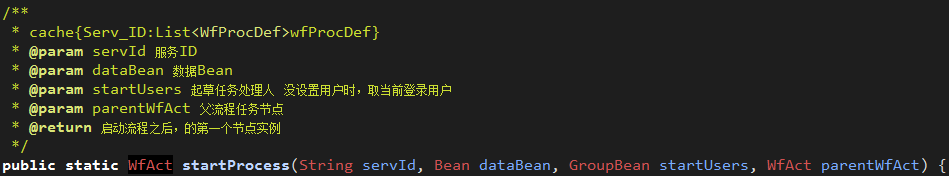


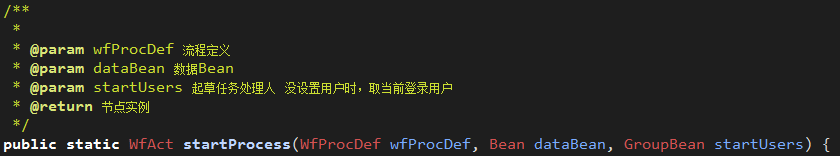
1. 上面的startWf方法调用的是服务SY\_WFE\_PROC\_DEF的start方法，所以，使用者也可以直接调用此服务方法

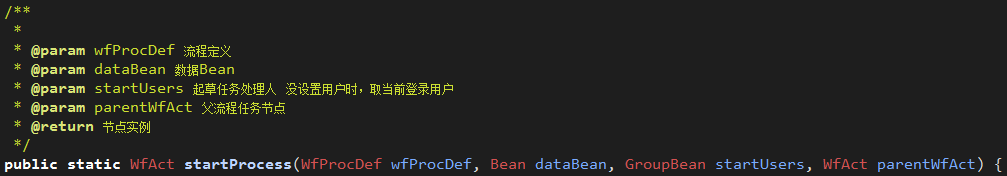


1. 调用WfProcessFactory中的startProcess方法









通过调用startProcess方法，我们可以使用服务按默认规则过滤的流程定义。同时还可以启动指定版本的流程。

* + - 1. 流程启动过滤规则

按默认规则过滤时，会启动哪个流程呢？

因为一个服务会走多种流程，每种流程又会有多个版本，那么，当不指定流程定义时，会启用哪个流程？

**首先，可使用的流程定义为**：

1. 流程定义必须启用，S\_FLAG=1；
2. 流程定义已生效，生效日期为空或早于当前时间，即EFFECTIVE\_DATE为空或小于当前时间；
3. 属于本公司的流程定义，S\_CMPY为本公司；

或为公共流程定义，S\_PUBLIC=1；

**然后，对这些流程定义进行排序，会启动排序最靠前的流程定义**：

1. 绑定排序BIND\_SORT升序，BIND\_SORT小的优先；
2. 同名流程定义的不同版本，版本号大的优先；
3. 本公司的流程定义优先；公共流程定义靠后；
4. 必须满足绑定条件，即BIND\_SCRIPT的条件。
   1. 流程监控

在流程运行的各环节，可以以列表形式、图形形式查看流程的运行。

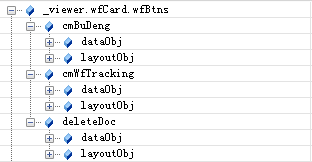
1. 二次开发
   1. 前台
      1. 前台结构

在前台的card页面，通过调试能很清楚地发现流程card的结构，主要属性、主要方法。

用\_viewer表示card对象，则\_viewer.wfCard包含了流程相关的一切，wfCard是在sy/base/frame/engines/rhWfCardView.js中定义的。



* + 1. 获取按钮对象



流程页面的按钮对象有两种：

1. 动能性按钮

主要包括 ①流程上配置的全局按钮

②通过“操作权限”配置的流程按钮

③通过“操作权限”配置的审批单按钮

④根据节点属性生成的按钮，如“办结”、“独占”按钮等

这些按钮存在wfBtns属性中。

获取方法（以“流程跟踪”按钮为例）：

\_viewer.wfCard.wfBtns["cmWfTracking"]

或 \_viewer.wfCard.\_getBtn("cmWfTracking")

1. 送交按钮

即通过连线生成的按钮，包括送交下一步的按钮、返回XXX的按钮。这些按钮存在wfNextBtns及wfNextBtnNames属性中。

获取方法（以“送领导审核”按钮为例）

根据CODE获取，CODE为连线的code：\_viewer.wfCard.\_getWfNextBtn(26)

根据NAME获取，\_viewer.wfCard.\_getNextBtnByName("送领导审核")

也可以通过属性来获取。

* + 1. 按钮对象的内容

按钮中含有2个对象：dataObj和layoutObj。

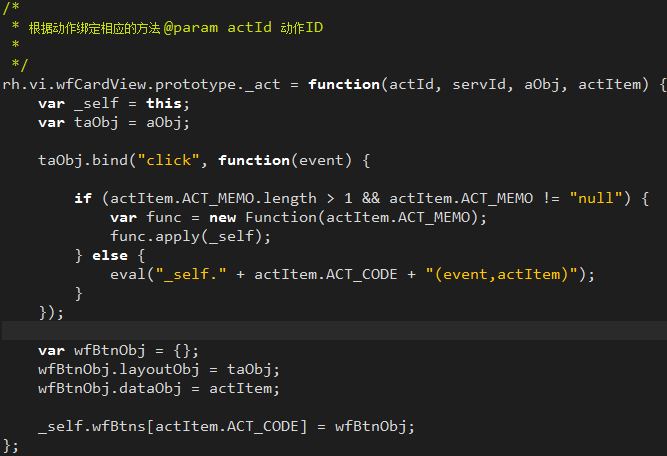
dataObj：按钮的ACT定义

layoutObj：页面上的元素对象，绑定事件是绑定在它上面

* + 1. 按钮对象的操作事件

1. 功能性按钮

参考代码：

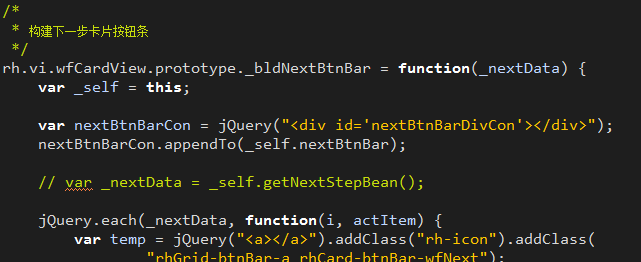


点击按钮后，会执行方法名为按钮的ACT\_CODE（从dataObj中可获取）的方法。

有些方法中会有一些扩展点，让使用者可以自定义，比较常用的有：

1. “办结”方法finish之前执行\_beforeFinish
2. “取消办结”方法undoFinish之前执行\_beforeUndoFinish
3. 送交按钮

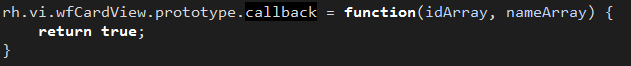
参考代码：



扩展点：

1. “返回XXX”按钮，可以扩展\_beforeSongjiao、callback方法。
2. 送交下一步按钮被点击时，主要的执行方法是\_openSelectOrg方法。

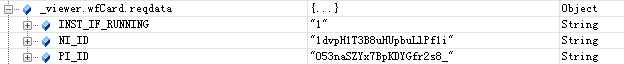
该方法进行处理人的选择，只有一个候选人时不弹出对话框；确定人员后执行\_confirmSend方法。扩展方法为\_beforeSongjiao、callback方法。\_beforeSongjiao在弹出选择框之前执行，callback在选择了人员后执行，callback方法的参数为选择的处理人的id及name。



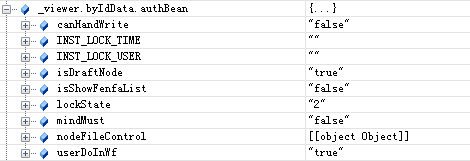
* + 1. 流程节点信息

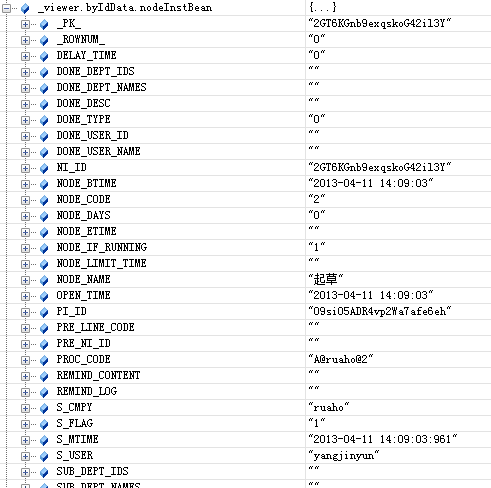
在任务处理界面，通常需要获取流程信息、任务节点信息。

1、通过属性获取：



界面上各种信息都可以通过byIdData获取：





2、通过方法获取

主要方法有getNodeInstBean、getAuthBean、getButtonBean等。

如：获取流程时候运行信息，\_viewer. byIdData.S\_WF\_STATE或\_viewer.wfCard.wfState。

* 1. 后台
     1. 处理人扩展

处理人扩展在配置上是下面这种格式：

com.cplh.pms.pj.ProjectPostUserBinder,,{'projectField':'PROJ\_ID','postCodes':'PROJ\_POST\_YIJIWENKONG','mode':'1','isCurUserProject':'false'}

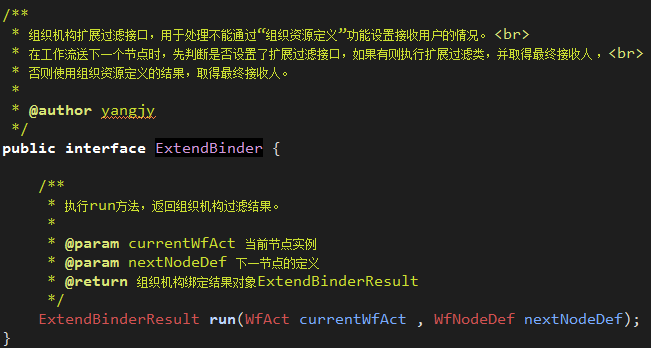
不需要参数的话，直接配置上扩展类即可：如com.cplh.pms.pj.ProjectPostUserBinder

需要参数，则在,,后面以json串的形式加上参数。

扩展类在实现上，有2种写法：

1. 实现接口com.rh.core.wfe.resource.ExtendBinder，返回的结果com.rh.core.wfe.resource.ExtendBinderResult。主要用来指定处理人所在的部门、角色等信息，构建一棵部门-角色-人员树，返回前台让用户进行选择。

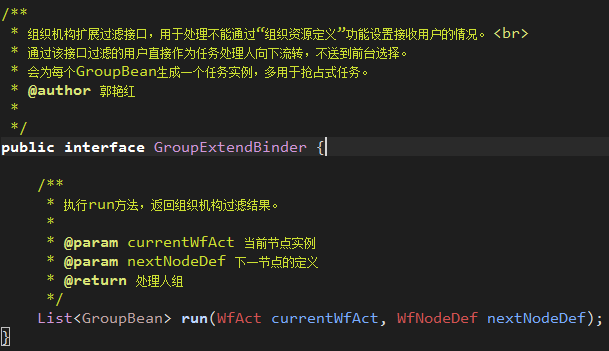
本质上，这种实现方式与我们通过配置部门、角色、用户信息产生的结果是一样的。



例子请参考：HuiQianExtWfBinder.java

1. 实现接口com.rh.core.wfe.resource.GroupExtendBinder，返回的结果为

List<GroupBean>。返回的结果不会返回到前台供用户选择，直接就进行了任务分配。

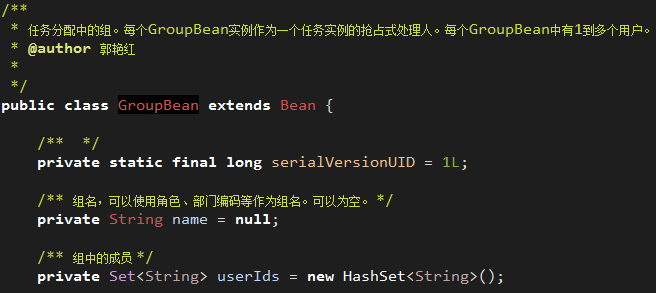
****

例子请参考EmployeeBinder.java。

**总结**：

不管是通过配置，还是用两种方式实现的处理人扩展，都是为了给创建的任务分配处理人。

各种配置和扩展最终再进行处理，最终的存在形式都是List<GroupBean>列表，列表的大小决定要创建的任务的个数，每个任务的处理人都是从GroupBean中获取的。GroupBean中有用户集合，集合中仅有一个用户时，任务分配给这个用户处理；集合中有多个用户时，任务在这些用户间进行抢占式处理。



1. 子流程配置及扩展

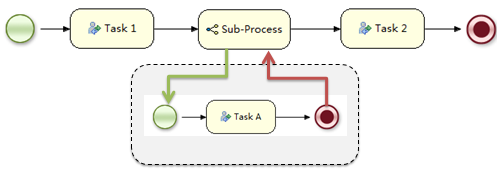
在使用子流程前，先了解子流程要解决的问题。

**子流程的需求**：

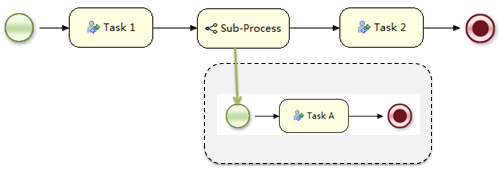
1. 流程运行中使用的表单不同，表单数据间有汇总——明细关系；
2. 流程运行中选定多组人，在组内进行业务处理，如抢占式处理；
3. 流程多个分支的同时执行；
4. 分支的运行（可能）不影响流程的运行。

**运行时的模型表示**：

1. 子流程的运行影响主流程的流转



1. 主子流程异步运行



**实现原理：**

假设，流程FlowA使用服务ServA，流程FlowB使用服务ServB，目前FlowA与FlowB之间没有任何关系。FlowA会触发FlowB的启动。

要使流程间建立主子服务关系，要在FlowA的节点中增加一个节点“子流程节点”，该节点上会实现对子流程FlowB的相关配置，包括起草人、启动策略等。将2个流程关联到一起。

需要注意的是：FlowB既可以作为FlowA的子流程由FlowA启动；同时也是独立的流程可自行启动。

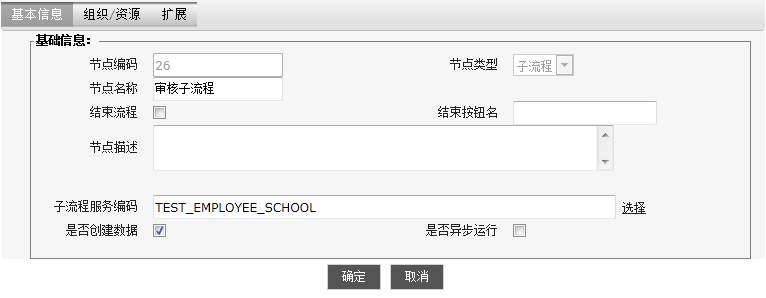
* 1. 配置

在流程节点配置中，有一个节点类型为“子流程”节点：。

子流程节点的配置主要有3方面内容。

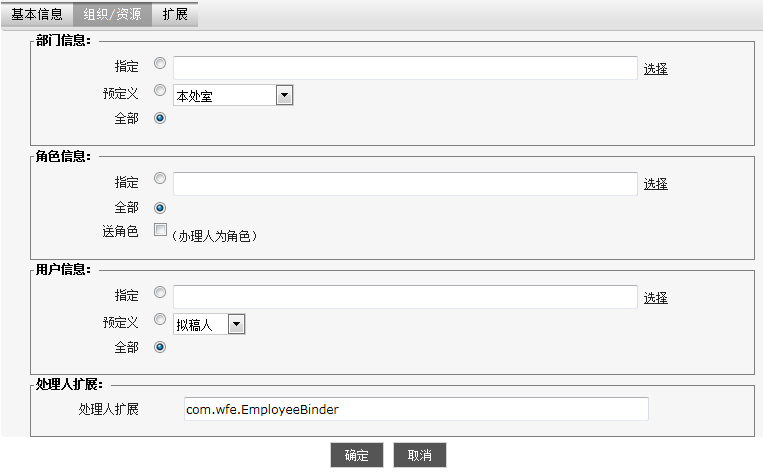
注：没有特别说明的配置项与普通节点的配置方式及作用是一样的。

* + 1. 基本信息



|  |  |  |
| --- | --- | --- |
| **配置项** | **使用说明** | **备注** |
| 子流程服务编码 | 要启动的子流程使用的服务。  运行到该节点时，会根据配置的服务，找到要启动的流程。具体会启动哪个流程，是按默认规则过滤出来的。  如果有业务需要启动特定一个流程的话，可以在“扩展”中进行配置 |  |
| 是否创建数据 | 子流程的服务数据是已经存在的还是需要运行时创建的，需要运行时创建的设置为true |  |
| 是否异步运行 | 启动子流程时，主流程运行不受干扰，设为true；子流程的运行影响主流程运行的，设为false |  |

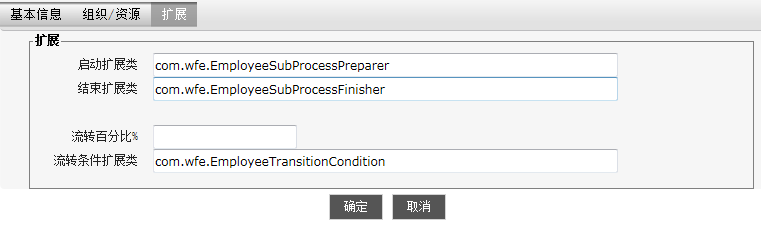
* + 1. 组织资源



这部分的配置跟普通节点的配置是一样的。

主要是配置子流程起草节点的处理人，也就是子流程的起草人。

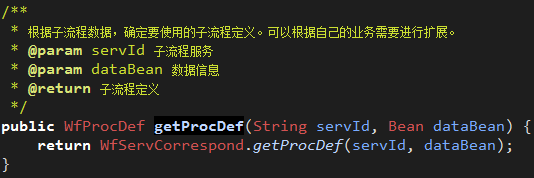
* + 1. 扩展



**启动扩展类**

要继承com.rh.core.wfe.serv.SubProcessPreparer类，其中有2个方法可能需要我们扩展：

1. 获取要启动的子流程定义，这里是按过滤规则取的默认的。



1. 获取子流程数据。默认是根据过滤规则进行获取，用户同样可根据业务需要扩展。



如：

@Override

**public** List<Bean> **prepareData**(List<GroupBean> startUsers) {

List<Bean> dataBeanList = **super**.prepareData(startUsers);

**for** (**int** i = 0; i < dataBeanList.size(); i++) {

Bean dataBean = dataBeanList.get(i);

dataBean.set("SCHOOL", startUsers.get(i).getUserNames());

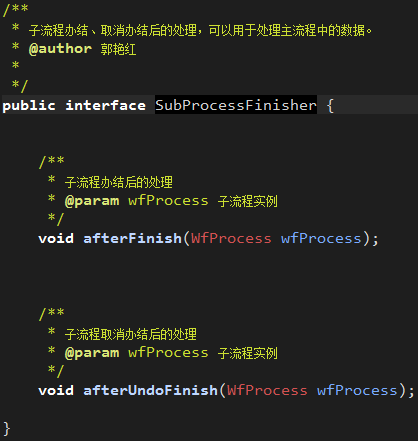
}

**return** dataBeanList;

}

**结束扩展类**

要实现com.rh.core.wfe.serv.SubProcessFinisher接口，实际业务中很少需要对此接口进行扩展。

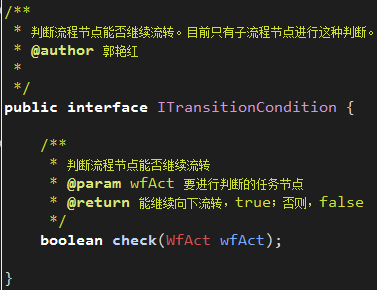


**流转百分比**

子流程启动后，如果“异步运行”设置为false，那么主流程就被阻塞了。“流转百分比”用来表明子流程（很可能是多个）运行完指定的百分比后，主流程就恢复运行；剩下的未办结的子流程继续运行。

如设为50后，表示50%的子流程办结后，主流程可继续运行。

**流转条件扩展类**

**流转百分比**其实是流转条件的一种简单实现，在具体业务中，需要更复杂的规则判断主流程在什么场景下可继续运行。需要实现com.rh.core.wfe.condition.ITransitionCondition接口：

* 1. 工作流程

基于一个场景来介绍子流程的启动、扩展。

准备：主流程FlowMain，服务为ServMain；子流程FlowSub，服务为ServSub。

主流程为：



子流程为



1. 主流程起草；
2. 起草人办理完成后，下一个节点是“审核子流程”，此时会弹出人员选择框，选择FlowSub起草节点的办理人；如果设置了“办理人扩展”，并且实现了GroupExtendBinder接口，就不会弹出选择框了。
3. 为FlowSub选择了处理人后，到后台进行处理，后台处理逻辑为：
4. 创建FlowMain的“审核子流程”的任务taskMain，这个任务的处理人就是上一节点的处理人，任务Id假定为taskInstanceId。
5. 接着判断主流程还能继续运行吗？能继续运行的条件为：主子流程异步运行 或 满足流程配置的流转条件。如果不能继续运行，会将taskMain暂停，主流程就不能继续处理了。
6. 启动子流程。默认实现都在com.rh.core.wfe.serv.SubProcessPreparer中。

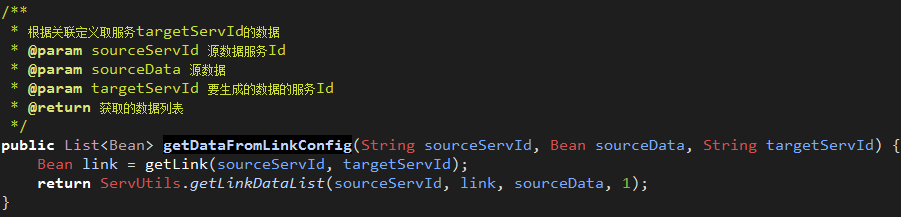
**首先要确定子流程的服务数据**。这里有几种情况：

1. 配置了启动扩展类并且覆盖了prepareData方法，将按照自己的方式准备数据；
2. 其他情况都是使用的系统默认提供的数据准备方式，要求主流程服务与子流程服务间必须有关联关系。

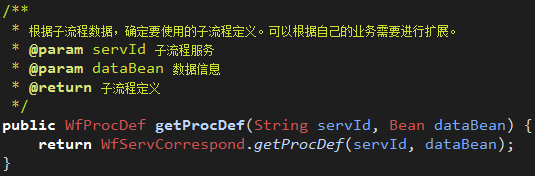
如果“是否创建数据”为true，表示此时要创建数据，创建子服务数据时，先调用子服务的byid初始化数据，然后根据关联关系初始化主服务传给它的数据，具体实现：



如果“是否创建数据”为false，则会根据关联服务查询子服务的数据：



**然后确定要启动的流程**。系统的默认实现：



同样可以根据需要扩展。

1. 子流程运行。
2. 子流程办结。办结时如果配置有“结束扩展类”，则会执行afterFinish方法，子流程每有办结，就会判断“流转条件”，满足条件时，主流程任务就被恢复，继续运行。
3. 子流程取消办结。如果配置有“结束扩展类”，则会执行afterUnFinish方法，也会判断“流转条件”，当发现不满足条件时，主流程任务就暂停。
4. 主流程继续运行。

**注，子流程取消办结的约束：**

1. 异步运行的情况下，子流程取消办结不受约束；
2. 主流程已经办结，子流程不能取消办结；
3. 主流程任务已经流转走了，子流程不能取消办结。
4. 其他细节

主要对流程运行设计的其他细节进行详细讲解。包括使用的表、使用的服务、待办、提醒、办理意见等。

* 1. 使用的服务

此处仅列出常用的服务及表。

**与流程定义相关：**

|  |  |  |
| --- | --- | --- |
| 服务名 | 表名 | 用途 |
| SY\_WFE\_PROC\_DEF | SY\_WFE\_PROC\_DEF | 流程定义 |
| SY\_WFE\_NODE\_DEF | SY\_WFE\_NODE\_DEF | 节点定义 |
| SY\_WFE\_NODE\_ACT | SY\_WFE\_NODE\_ACT | 流程节点和ACT关联信息 |
| SY\_WFE\_LINE\_DEF | SY\_WFE\_LINE\_DEF | 节点连线定义 |
| SY\_WFE\_CUSTOM\_VAR | SY\_WFE\_CUSTOM\_VAR | 节点自定义变量 |

**与流程运行（流程实例）相关：**

|  |  |  |
| --- | --- | --- |
| 服务名 | 表名 | 用途 |
| SY\_WFE\_PROC\_INST | SY\_WFE\_PROC\_INST | 流程实例 |
| SY\_WFE\_PROC\_INST\_HIS | SY\_WFE\_PROC\_INST\_HIS | 流程实例历史信息 |
| SY\_WFE\_NODE\_INST | SY\_WFE\_NODE\_INST | 节点实例 |
| SY\_WFE\_NODE\_INST\_HIS | SY\_WFE\_NODE\_INST\_HIS | 节点实例历史信息 |
| SY\_SERV\_FLOW | SY\_SERV\_FLOW | 流经表 |

注：流经记录是在打开任务的时候才产生的。起草节点也会有流经记录。

**与任务分配相关：**

|  |  |  |
| --- | --- | --- |
| 服务名 | 表名 | 用途 |
| SY\_WFE\_NODE\_USERS | SY\_WFE\_NODE\_USERS | 任务分配表 |

**与待办已办相关：**

|  |  |  |
| --- | --- | --- |
| 服务名 | 表名 | 用途 |
| SY\_COMM\_TODO | SY\_COMM\_TODO | 待办事务 |
| SY\_COMM\_TODO\_HIS | SY\_COMM\_TODO\_HIS | 已办事务 |

**与办理意见相关：**

|  |  |  |
| --- | --- | --- |
| 服务名 | 表名 | 用途 |
| SY\_COMM\_MIND\_TYPE | SY\_COMM\_MIND\_TYPE | 意见类型 |
| SY\_COMM\_MIND\_CODE | SY\_COMM\_MIND\_CODE | 意见信息 |
| SY\_COMM\_MIND | SY\_COMM\_MIND | 办理意见 |

**与分发相关：**

|  |  |  |
| --- | --- | --- |
| 服务名 | 表名 | 用途 |
| SY\_COMM\_SEND | SY\_COMM\_SEND | 分发方案表 |
| SY\_COMM\_SEND\_ITEM（父服务） | SY\_COMM\_SEND\_ITEM | 分发方案的明细 |
| SY\_COMM\_SEND\_ITEM\_DEPT | SY\_COMM\_SEND\_ITEM | 分发方案明细——部门 |
| SY\_COMM\_SEND\_ITEM\_ROLE | SY\_COMM\_SEND\_ITEM | 分发方案明细——角色 |
| SY\_COMM\_SEND\_ITEM\_USER | SY\_COMM\_SEND\_ITEM | 分发方案明细——用户 |
|  |  |  |
| SY\_COMM\_SEND\_DETAIL |  | 分发明细（直接选择人、部门、角色进行分发时用） |

**与ENTITY相关：**

|  |  |  |
| --- | --- | --- |
| 服务名 | 表名 | 用途 |
| SY\_COMM\_ENTITY | SY\_COMM\_ENTITY | 记录表，维护每个服务的每条数据的信息,如创建信息、删除信息。仅当服务设置了“标题格式”时，该服务的数据才维护到此表中。数据修改时，也会同步实体表信息。  数据显示时，仅显示用户自己创建及有权限查询（同机构）的数据。 |
|  |  |  |
| SY\_COMM\_ENTITY\_STATE（父服务） | SY\_COMM\_ENTITY | 记录表。无过滤。 |
| SY\_COMM\_ENTITY\_DOING |  | 待办记录表。 |
| SY\_COMM\_ENTITY\_DONE\_FINISH |  | 流经本人的并且流程已经办结的记录。查询了SY\_SERV\_FLOW表。 |
| SY\_COMM\_ENTITY\_DONE\_FINISH\_MINE |  | 本人起草的记录，流经本人的并且流程已经办结的记录。  查询了SY\_SERV\_FLOW表。 |
| SY\_COMM\_ENTITY\_DONE\_RUN |  | 任务曾经分配给本人，并且该任务已办理完成。 |
| SY\_COMM\_ENTITY\_DONE\_RUN\_MINE |  | 本人起草的记录，任务曾经分配给本人，并且该任务已办理完成。 |
| SY\_COMM\_ENTITY\_ZHUBAN |  | 本人起草的记录。 |
| SY\_COMM\_ENTITY\_ZHUBAN\_FINISH |  | 本人起草的并且流程已办结的记录。 |
| SY\_COMM\_ENTITY\_ZHUBAN\_RUN |  | 本人起草的并且流程未办结的记录。 |

注：流经本人的意思是：流程流转过程中，有任务送交给本人，并且本人已经打开过该任务。与SY\_SERV\_FLOW表有关。曾经分配给本人的意思是：流程流转过程中，有任务送交给本人。与sy\_wfe\_node\_users 表有关。

* 1. 流程定义

流程定义的信息主要通过flash进行绘制；每个流程由点、线组成。分别存在不同的表。表的细节请自己参考数据库。

* 1. 流程运行

流程实例可以通过自动启动或手动启动的方式进行创建。

流程实例创建后，马上会创建一个起草任务。图示如下：

**运行中：**

**办结：**

流转场景

|  |
| --- |
| **启动** |
| 1、创建流程实例；  2、创建起始任务；  新增起始节点的实例；  增加待办  3、创建主办事务 |
| **向下送交** |
| A流转到B：结束当前任务，将流程流到下面的节点  finish(A)  将待办任务的状态置为已办；  将节点实例的状态置为已处理；  toNext(B)  创建下一节点任务：包括节点实例和待办任务 |
| **收回** |
| A—>B—>C，B发给了C，在C处理之前可收回。  finish(C) toNext(B)。如果B发送给了多个节点，一收回，将所有的节点都收回了。 |
| **办结** |
| 执行process.finish()  1、流程实例结束、节点实例结束 、待办结束  2、将上面的记录移到相应的历史表中 |
| **取消办结** |
| 执行process.undoFinish()  执行跟“办结”相反的操作 |
| **并发** |
| 创建多个节点实例，待办任务 |
| **合并** |
| A为并发节点，B为合并节点 A(fork)—>B(join)  A有多个任务，A中每个任务办理后，都会为B创建一个任务，紧接着，由于B是合并节点，B上的任务会进行合并，仅仅保留一个任务实例，之前创建的B的任务实例都被  Finish了。  在流程跟踪会显示：这个合并节点被finish了n次。 |

* 1. 待办已办

每次为用户创建一条新任务时,都会为任务的每个待办理人创建一条待办记录，都会在SY\_COMM\_TODO中创建一条新的待办。待办表中主要的字段如下：

|  |  |
| --- | --- |
| **字段名** | **含义** |
| SEND\_USER\_CODE | 待办任务的发送人（创建者） |
| OWNER\_CODE | 需要进行待办办理的用户 |
| DEPT\_CODE | OWNER用户所在的dept |
| TDEPT\_CODE | OWNER用户所在的tdept |
| SERV\_ID | 服务Id |
| SERV\_DATA\_CODE | 服务定义中的“SERV\_DATA\_CODE”进行数据替换后的内容 |
| S\_EMERGENCY | 任务紧急程度。具体取值参考数据字典SY\_COMM\_TODO\_EMERGENCY |
| TODO\_FROM | 待办来自哪种业务：如wf工作流，remind消息提醒。一般直接赋值为wf |
|  | |
| TODO\_TITLE | 待办标题，可以取SERV\_DATA\_CODE的值。 |
| TODO\_URL | 待办链接。如TEST\_EMPLOYEE\_FLOW.byid.do?data={\_PK\_:2FPN9jvWFdJWfyH\_fQ\_sKZ, NI\_ID:0UH70bs7F9EVy1Ae269FIv}。  格式为：服务.byid.do。data参数中的\_PK\_为服务的数据的主键。NI\_ID为任务实例（SY\_WFE\_NODE\_INST）的Id。  如果是自己手动添加一些待办数据（不走流程），则TODO\_URL的格式用户可自定义，并在前台自行解析处理 |
| TODO\_SEND\_TIME | 待办创建时间 |
| TODO\_FINISH\_TIME | 待办完成时间 |
| TODO\_CODE | 待办类型编码，通常设置为服务Id |
| TODO\_CODE\_NAME | 待办编码名称，通常设置为服务名称 |
| TODO\_OPERATION | 待办操作，设置为节点名称或任务名称都可 |
| TODO\_CATALOG | 待办分类，1、待办 ；2、待阅 |
| TODO\_OBJECT\_ID1 | 服务的数据Id |
| TODO\_OBJECT\_ID2 | 分发时设置为分发Id；其他情况设置为待办任务实例Id |
| TODO\_CONTENT | 待办提示消息内容；消息提醒时有用 |

其中，比较重要的字段有TODO\_URL、TODO\_CATALOG。

**紧急字段**S\_EMERGENCY

要在待办中显示“紧急”字段，业务表单中必须有S\_EMERGENCY字段。它的值会赋给待办服务SY\_COMM\_TODO的S\_EMERGENCY字段。否则，S\_EMERGENCY的值默认都是“一般”。

**待办分类**TODO\_CATALOG

待办分类的原因：有些任务必需要用户处理，否则可能影响流程向下流程，叫“待办”任务；有些任务，如分发任务，用户阅知即可，叫“待阅”任务。

不同任务，可以进行不同处理，如“待阅”任务，用户一进行查看，该任务就办理完成了。

**\* SY\_COMM\_TODO表中待办的生成规则**

1、普通任务，该节点生成一个新任务，待办理人只有一个，生成的待办也只有1条；

2、并发式任务，该节点生成多个新任务，每个新任务有一个待办理人，会为每个待办理人生成1条待办；

3、抢占式任务（不并发），该节点生成一个新任务，但有多个待办理人，会为每个待办理人生成1条待办；但是，当其中一个办理人点击“开始处理文件”按钮将该任务归为己有后，与他抢占同一任务的待办理人的待办就消失了。

4、抢占式并发任务，通常需要实现com.rh.core.wfe.resource.GroupExtendBinder接口来实现。会生成多个任务，每个任务有多个待处理人，每个待处理人都会收到待办，类似简单的抢占式任务。

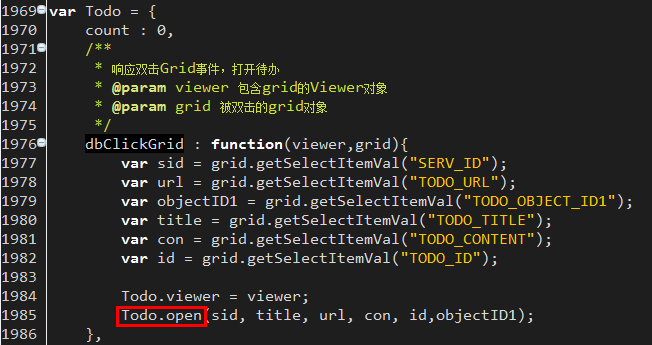
在项目中，可能需要对待办的展示进行扩展，主要有以下几方面需要注意：

\***（服务形式的）待办列表**

在待办扩展js（SY\_COMM\_TODO\_list.js）中，打开待办事务的方法为：

Todo.dbClickGrid(\_viewer,\_viewer.grid);

该方法在tools.js中



请仔细阅读open方法的代码，这段代码写明怎样打开一条待办进行任务处理的。

\***（首页形式的）待办**

通常，项目首页上会配置上“待办”“已办”区块。

前面已经讲过，首页已办通过“门户模板”进行配置，首页上的各个区块则是“模板组件”。

“待办”的模板组件，平台中已经预制了一个“SY\_CM\_TODO”，对应的ftl文件为SY\_COMM\_TODO.ftl。

打开SY\_COMM\_TODO.ftl，可以看到ftl中打开待办进行处理的代码为



模板组件的扩展处理代码都是在门户js中处理的。因此openTODOCard在/sy/comm/home/js/incl-templ.js中有处理，见文件最前面的位置。

可以看到todoCatalog=2（待阅）时，一打开任务页，该任务就办理完成了，待办列表中就没有了。

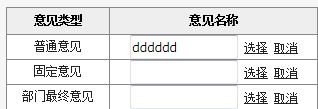
**注意**：openTODOCard方法与前面讲的Todo.open方法作用相同。因此，在使用时，一定要注意这两处的代码逻辑是一样的。

如果用户自己对“待办”表进行数据维护，需要对打开“待办”任务的代码进行扩展时，要注意同时处理这两个地方。

* 1. 办理意见

办理意见通过节点进行配置。

在配置意见时，可以配置3种类型的意见：



意见的渲染过程：

rh.vi.cardView.prototype.\_afterLoad （rhCardView.js）

rh.vi.wfCardView.prototype.render （rhWfCardView.js）

rh.vi.wfCardView.prototype.initMind

rh.cm.mind.prototype.render （/sy/comm/mind/mind.js）

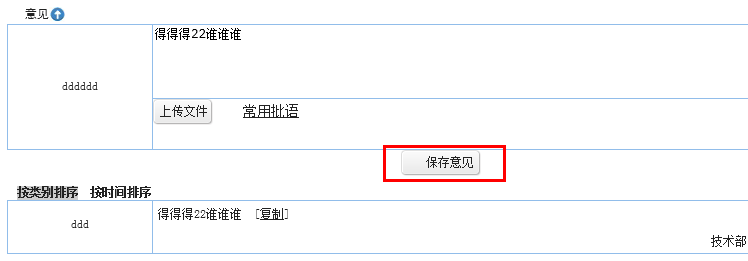
在render方法中，会取获取意见数据，获取意见显示模板，构建前台已经输入框，显示已经汇总的意见，将意见按时间、类型进行排序显示等。

1. 意见标题模板文件 /sy/comm/mind/mindListTitle.ftl
2. 意见列表模板文件

在mindListTitle.ftl中引用了/sy/comm/mind/mindList.ftl进行意见列表显示，显示时可以按时间排序（mindListByTime.ftl），或者按类别排序（mindListByType.ftl）

1. 意见输入框模板文件 /sy/comm/mind/mindInput.ftl

在意见输入框下方，有一个“保存意见”按钮。



按钮主要作用是，保存意见后，对页面进行刷新，使得刚刚保存的意见能即时出现在下方列表中。使用者觉得不需要的话，可将之从意见输入框模板ftl中移除即可。

1. FAQ
2. **流程使用的业务表单有什么要求**？

参考本文档***1.2.1节***。

1. **业务表单中的流程字段的含义**？

|  |  |
| --- | --- |
| **字段名** | **含义** |
| S\_WF\_INST | 流程实例 |
| S\_WF\_NODE | 当前节点，通常可以使用该字段来设置业务表单的状态。如在beforeSave方法中：  **if**(!paramBean.getStr("S\_WF\_NODE").isEmpty()){  paramBean.set("S\_WF\_STATUS",paramBean.getStr("S\_WF\_NODE").split(Constant.*SEPARATOR*)[0]);  } |
| S\_WF\_STATE | 流程状态，0：未在流程中 1：在流程中 2：流程已办结 |
| S\_WF\_USER | 当前节点办理人 |

1. **如何在流程办结后执行特定操作**？

流程流转过程中，会执行服务的save方法来更新流程状态相关的字段。而此时，S\_WF\_STATE的值为2，因为可以在beforeSave方法中这样写：

**if**(paramBean.getInt("S\_WF\_STATE") == WfeConstant.*WFE\_PROC\_INST\_NOT\_RUNNING*){

paramBean.set("S\_WF\_STATUS", "已办结");

paramBean.set("GW\_END\_USER", Context.*getUserBean*().getId());

paramBean.set("GW\_END\_TIME", DateUtils.*getDatetime*());

}