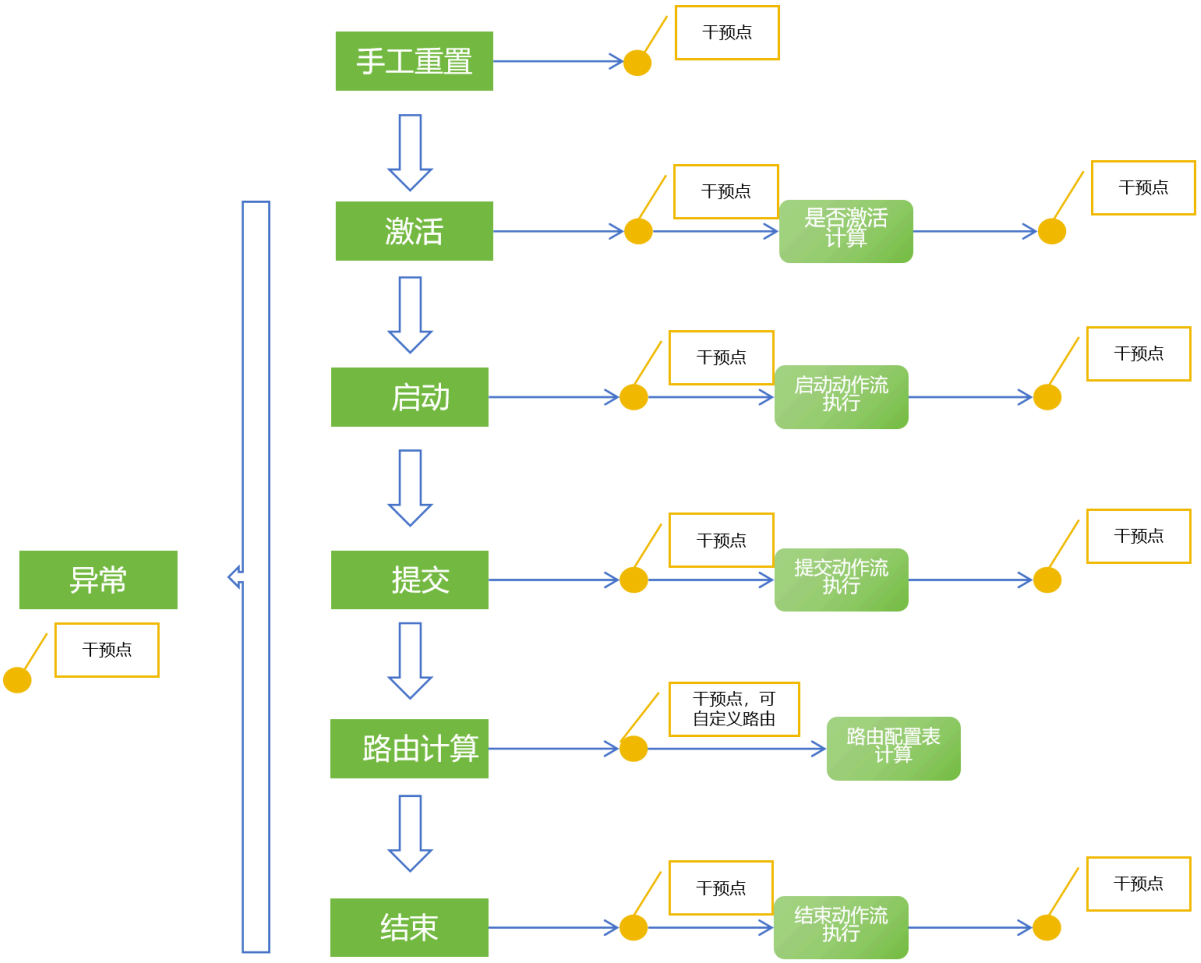
GPF节点扩展开发说明

1.目标

通过扩展节点的默认处理逻辑，封装好适合限定场景下的节点类型。

2.节点处理过程及开放干预点



3.节点扩展开发

通过实现DCNodeExtBehavior,DCNodeExtBehaviorUIIntf接口，并在节点上配置扩展类型，以循环头节点举例：

循环头节点的默认行为包括：

1. 在循环索引未初始化时要进行初始化；
2. 循环头不需要外部输入交互，自动提交下一步；
3. 循环头的所有下游节点是循环头节点结束时要前往的节点；

```
package cell.gpf.dc.basic.node.behavior;

import com.kwaidoo.ms.tool.CmnUtil;
```

```
import cell.CellIntf;
import cell.gpf.dc.runtime.IDCRuntimeContext;
import cmn.anotation.ClassDeclare;
import cmn.util.TraceUtil;
import cmn.util.Tracer;
import fe.cmn.res.FeIcons;
import gpf.dc.basic.node.behavior.DCNodeExtLoopValidator;
import gpf.dc.config.PDC;
import gpf.dc.config.RefPDCNode;
import gpf.dc.fe.dto.FlowNodeStyle;
import gpf.dc.fe.intf.DCNodeExtBehaviorUIIntf;
import gpf.dc.intf.node.DCNodeExtBehavior;
import gpf.dc.intf.node.DCNodeExtValidator;
import gpf.dc.runtime.PDCForm;
import gpf.dto.cfg.runtime.RouterOption;
import gpf.dto.model.cfg.NodeOption;
@ClassDeclare(label = "循环头"
,what="循环头节点，自动跳转路由，并根据指定的索引、遍历值属性"
, why = ""
, how = ""
,developer="陈晓斌"
,version = "1.0"
,createTime = "2025-03-27"
,updateTime = "2025-03-27")
public interface INodeLoopHeadBehavior extends CellIntf,
DCNodeExtBehavior,DCNodeExtBehaviorUIIntf{

    public final static String sIndexField = "索引属性";
    public final static String sLoopDataField = "遍历值属性";
    //----------------------------------界面展示相关实现----------------------------
-----------
    /**
     * 设置在流程图上的节点样式
     */
    @Override
    default FlowNodeStyle setFlowNodeStyle(FlowNodeStyle nodeStyle) throws
Exception {
        nodeStyle.setIcon(FeIcons.loop);
        return nodeStyle;
    }
    //------------------------------------------------------------------------------
-------------

    //----------------------------------后台运行逻辑相关实现------------------------
------------
    /**
     * 构建节点的运行参数：
     * 循环头默认跳转到下一步，所以不需要流程上的路由配置，开启自主路由和自动下一步的配置
     * 同时节点需要在启动时初始化遍历索引值，所以在启动节点时进行干预
     */
    @Override
    default NodeOption buildNodeOption(PDC pdc) throws Exception {
        return new NodeOption().setSelfRouting(true) //自主路由
                .setAutoGoNext(true) //自动下一步
                .setInterveneStart(true) //干预节点启动
```

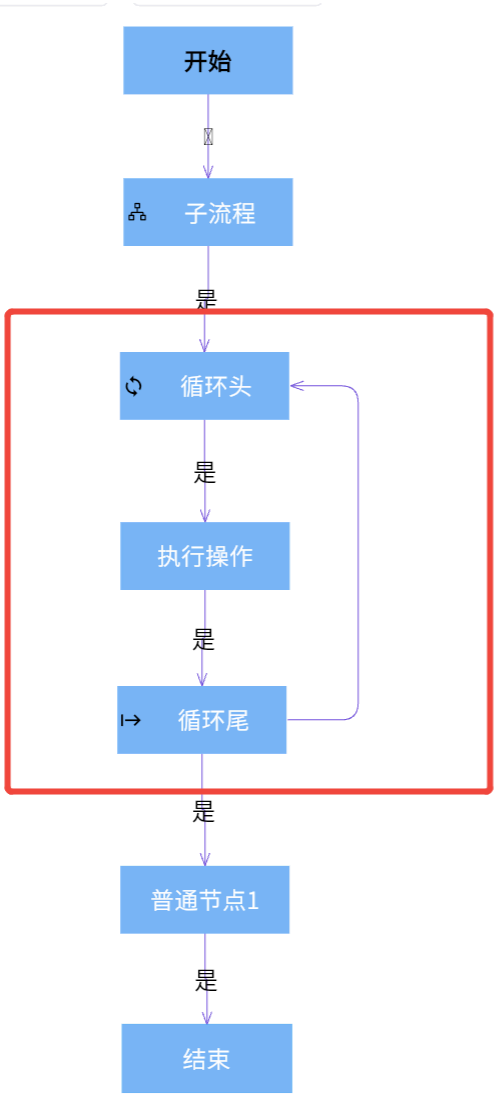
```
                ;
    }
    /**
     * 设置流程图保存时的校验器，循环头和循环尾必须成对出现，且循环体内节点不能与循环体外节点有
连线
     */
    @Override
    default Class<? extends DCNodeExtValidator> getValidatorClass() {
        return DCNodeExtLoopValidator.class;
    }

    /**
     * 构建循环头的路由规则，所有下游节点可有路由
     */
    @Override
    default RouterOption buildRouterOption(IDCRuntimeContext rtx) throws
Exception {
        //循环头节点，找到所有下游接口作为离开节点
        RouterOption option = new RouterOption();
        option.setGoNextAll(true);
        return option;
    }
    /**
     * 在重置循环头节点时，需要在上下文参数中传入设置循环重新初始化
     */
    @Override
    default void onRevoke(IDCRuntimeContext rtx) throws Exception {
        //重置节点时，将循环重置标识设置为false
        String loopInitedKey = getLoopInitedKey(rtx.getRefPDCNode());
        rtx.setParam(loopInitedKey, false);
    }
    /**
     * 在节点启动时，运行节点动作流前，判断是否需要重新初始索引值，并进行索引值初始化
     */
    @Override
    default void beforeRunStartAction(IDCRuntimeContext rtx) throws Exception {
        String initLoopKey = getLoopInitedKey(rtx.getRefPDCNode());
        boolean loopInited = CmnUtil.getBoolean(rtx.getParam(initLoopKey),false);
        Tracer tracer = TraceUtil.getCurrentTracer();
        PDCForm pdcForm = rtx.getPdcForm();
        String indexField = getIndexField(rtx);
        Long value = pdcForm.getLong(indexField);
        tracer.info("索引="+value);
        //循环索引未初始化的，先初始化
        if(!loopInited) {
            pdcForm.setAttrValue(indexField, 0L);
            rtx.setPdcForm(pdcForm);
            rtx.setParam(initLoopKey, true);
            tracer.info("初始化循环索引");
            tracer.info("isPdcFormModified:"+rtx.isPdcFormModified());
        }
    }
    //-------------------------------------------------------------------------
-------------
```

```
    //--------------------------------获取参数的辅助方法-------------------------
----------
    /**
     * 获取PDC上配置的索引属性参数
     * @param rtx
     * @return
     * @throws Exception
     */
    default String getIndexField(IDCRuntimeContext rtx) throws Exception {
        PDC pdc = rtx.getPdc();
        String indexField = pdc.getString(sIndexField);
        if(CmnUtil.isStringEmpty(indexField)) {
            throw new Exception("索引属性未配置！");
        }
        return indexField;
    }
    /**
     * 获取PDC上配置的遍历值属性参数
     * @param rtx
     * @return
     * @throws Exception
     */
    default String getLoopDataField(IDCRuntimeContext rtx) throws Exception {
        PDC pdc = rtx.getPdc();
        String loopDataField = pdc.getString(sLoopDataField);
        if(CmnUtil.isStringEmpty(loopDataField)) {
            throw new Exception("遍历值属性未配置！");
        }
        return loopDataField;
    }
    /**
     * 获取循环是否已初始化的上下文参数key
     * @param node
     * @return
     * @throws Exception
     */
    public static String getLoopInitedKey(RefPDCNode node) throws Exception {
        return node.getKey() + "@loopInited";
    }
}
```

循环头节点上配置扩展类型和参数

4.使用效果示例



循环体配置

操作记录

当前阶段	状态	操作人	执行操作	操作时间	上一阶段	上一阶段...	接收人	发起人	创建时间	操作
普通节点1	todo				循环尾	超级管理员		超级管理员	2025-04-03	
循环尾	finish	超级管理员		2025-04-03	执行操作	超级管理员		超级管理员	2025-04-03	
执行操作	finish	超级管理员	下一步	2025-04-03	循环头	超级管理员		超级管理员	2025-04-03	
循环尾	revoke	超级管理员		2025-04-03	执行操作	超级管理员		超级管理员	2025-04-03	
循环头	finish	超级管理员		2025-04-03	循环尾	超级管理员		超级管理员	2025-04-03	
执行操作	finish	超级管理员	下一步	2025-04-03	循环头	超级管理员	[redacted]	超级管理员	2025-04-03	
循环尾	revoke	超级管理员		2025-04-03	执行操作	超级管理员		超级管理员	2025-04-03	
循环头	finish	超级管理员		2025-04-03	循环尾	超级管理员		超级管理员	2025-04-03	
执行操作	finish	超级管理员	下一步	2025-04-03	循环头			超级管理员	2025-04-01	
循环头	finish			2025-04-01	子流程			超级管理员	2025-04-01	
子流程	finish				开始	超级管理员		超级管理员	2025-04-01	
子流程	revoke				开始	超级管理员		超级管理员	2025-04-01	
开始	finish	超级管理员	下一步	2025-04-01				超级管理员	2025-04-01	

共13条 1 20条/页

5.其他

5.1 节点运行参数（NodeOption）说明：

参数	标签	说明
selfRouting	自主路由	true/false，启用后流程上配置的路由策略将无效，由节点自行实现
autoGoNext	自动下一跳	true/false，启用后将忽略节点交互动作做自动路由
saveTotalForm	保存总表单	true/false，节点执行时是否保存总表单
reQueryTotalForm	重查总表单	true/false，是否重新查询总表单数据，主要用于节点必须拿到最新总表单数据
enableSubFlow	启用子流程	开启后，将调用子流程，并在子流程完成时回调父流程
subFlow	子流程模型ID	
subAction	子流程提交动作	填写子流程开始节点的提交动作
interveneActive	是否干预激活环节	true/false，开启后，在节点激活方法执行前后可执行自定义干预逻辑
runAcitveAction	是否运行激活动作	true/false，关闭后将不执行进入路由激活动作，默认开启
interveneStart	是否干预启动环节	true/false，开启后，在节点启动动作流执行前后可执行自定义干预逻辑
runStartAction	是否运行启动动作	true/false，关闭后将不执行进入节点启动动作流，默认开启
interveneSubmit	是否干预提交环节	true/false，开启后，在节点提交动作流执行前后可执行自定义干预逻辑
runSubmitAction	是否运行提交动作	true/false，关闭后将不执行进入节点提交动作流，默认开启
interveneFinish	是否干预结束环节	开启后，在节点节点动作流执行前后可执行自定义干预逻辑
runFinishAction	是否运行结束动作	true/false，关闭后将不执行进入节点结束动作流，默认开启

5.2 路由参数（RouterOption）说明：

参数	标签	说明
goNextAll	离开路由包含所有下游节点	true/false,优先级高于nexts，默认false

参数	标签	说明
nexts	下一步节点Key列表	
resetBefore	跳转下一步前重置状态的节点列表	
resetAfter	跳转下一步后重置状态的节点	需要注意的是：NodeOption如果开启了autoGoNext时，自动提交为同步操作时重置节点不能包含自身，否则重置无效