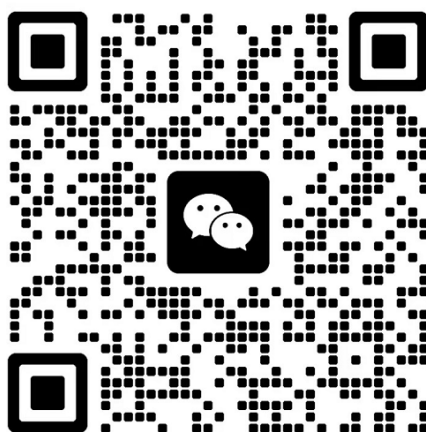


Easy WorkFlow workflow引擎使用指南

文档最后编辑日期:2024-02-19

作者 兔老三



扫一扫上面的二维码图案，加我为朋友。

目录

一、下载并开启引擎.....	4
下载方法.....	4
开启引擎.....	4
二、设计思想与名词解释.....	5
节点.....	6
任务节点.....	6
开始(根)节点.....	6
混合网关.....	7
结束节点.....	8
变量.....	8
表达式.....	9
事件.....	9
定时任务.....	10
流程实现示例.....	10
三、主要 func 说明	11
流程定义相关.....	11
●流程 save/update	11
●获取某个来源下所有的流程信息	11
●获取特定流程的定义	12
流程实例相关.....	13
●流程开始	13

•流程实例撤销	13
•获取特定流程实例信息	14
•特定流程实例中审批记录	14
•获取 “由我发起 ”的流程列表	14
任务相关.....	16
•通过	16
•驳回	16
•将任务转交给其他人	17
•待办任务列表	17
•已办任务列表	18
•获取特定任务信息	18
•获取本任务所在节点的所有上游节点	19
•自由驳回	19
•列出当前任务可以执行哪些操作	20
四、数据库表结构说明.....	21
流程定义资源表.....	21
流程节点关系表.....	21
流程实例表.....	22
流程实例中变量定义表.....	22
流程任务表.....	23
五、源码文件结构说明.....	24

一、下载并开启引擎

下载方法

方法一

在 go.mod 文件中添加

```
github.com/Bunny3th/easy-workflow latest
```

而后运行 `go mod download` 命令即可

方法二

```
go get github.com/Bunny3th/easy-workflow@latest
```

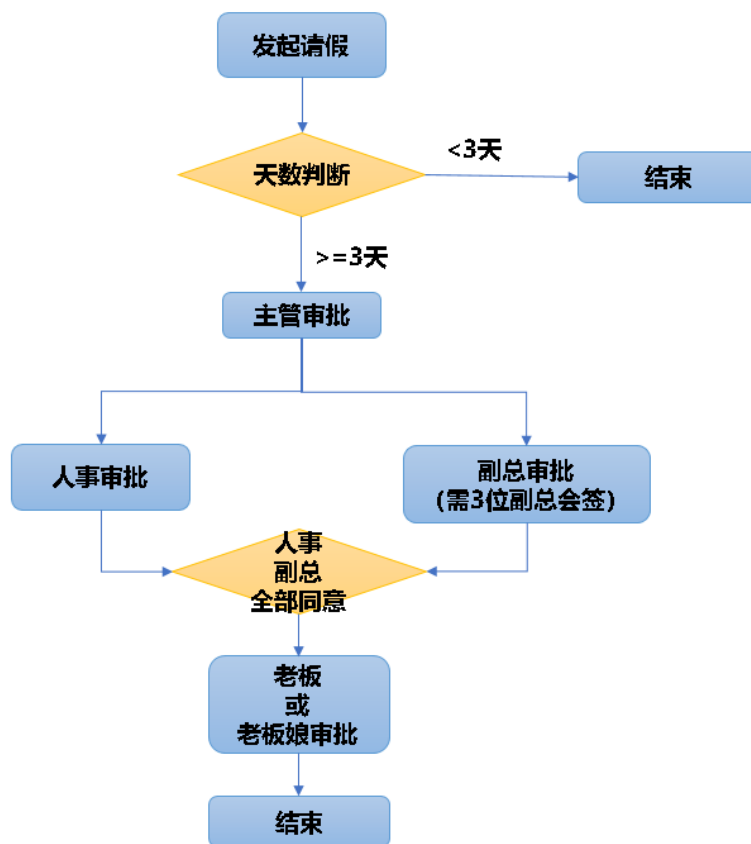
开启引擎

请查看 [easy-workflow/example/main.go](#) 中示例代码

二、设计思想与名词解释

在作者看来，所有的流程设计，均是“以节点流转体现，以任务分配落实”（*请注意，这句话贯穿本项目始终）。

在源码 example 文件夹下，定义了一个颇为奇葩的请假流程：



在 BPMN(Business Process Modeling Notation 业务流程建模标注)规范中，流程各个节点之间是由“连接对象”（在流程图中表现为“连接线”）形成关系连接。但本项目作者比较懒且奇葩，直接使用链表方式处理关系。故而在本项目中，流程定义只依赖一种对象：“节点”。

所以，注意上图的流程在本项目的实现中，不存在“连接对象”。“发起请假”是一个节点，“天数判断”亦是一个节点。他们之间的关系，是由“天数判断”节点中“上级节点 ID”所确定的。

本项目中节点关系由链表实现，无“连接对象”，这是重点。

节点

节点的作用是定义、改变流程走向。（数据结构定义在 easy-workflow/workflow/model/Node.go 文件中）

项目中有 4 种节点：

任务节点

流程需要以任务分配作为落实，如上图“请假流程”，无论节点间如何流转，最终落实的是“任务”。

任务节点决定了“何时将什么任务分配给哪些人”，流程流转到哪个节点，节点中的分配人就需要完成“通过或驳回”的对应审批任务。

开始(根)节点

可以把开始节点看作是一个特殊的任务节点。其特殊点体现在：

A、一个流程只能有一个开始节点

B、开始节点中的任务“开始即完成”。这一设计是借鉴了 activiti 引擎的不便之处。

java 程序员常用的 activiti 引擎中，一个简单的工作流示例如下：



在 activiti 的设计中，工作流开始后会生成一个“学生”节点的任务，任务状态为“待办”。这一点困扰了很多，百度上一大片“activiti 启动流程并完成第一个任务”的问题。从人类思维逻辑上讲，“学生”是申请者，其本意就是希望节点流转到下一个“班主任”节点。而 activiti 中却还要再完成“学生”节点的初始任务（在 activiti 中，往往需要使用事件完成此任务）。

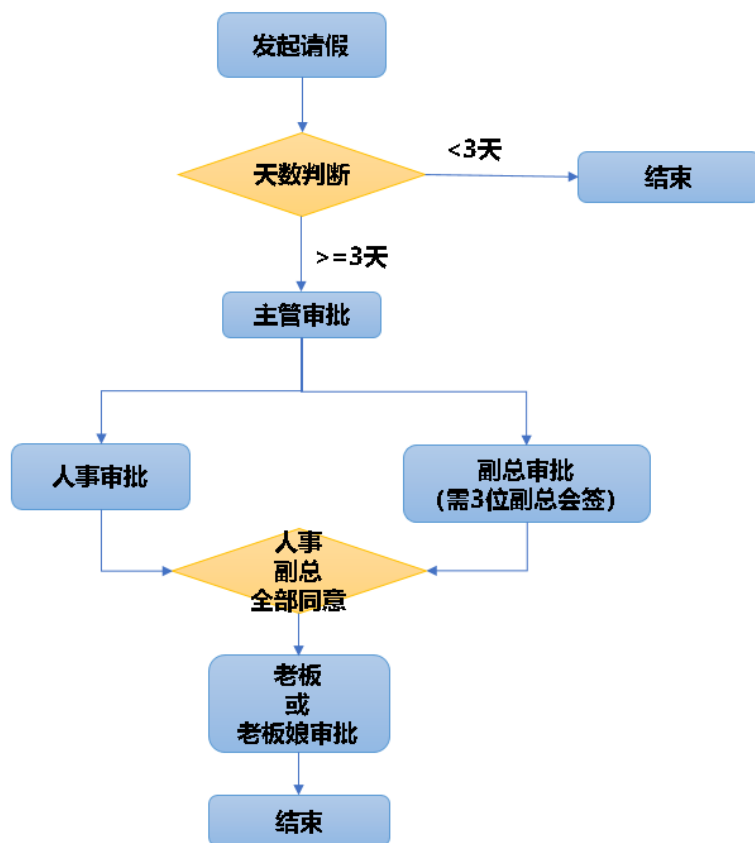
所以在本项目中，开始节点产生的任务自动完成。

混合网关

本项目中的混合网关，是借鉴了 activiti 中排他、并行、包含等网关以及 BPMN 规则中“连接线”的部分功能，故名“混合网关”。其本质上起到的作用就是“条件分支选择”。

（数据结构定义在 easy-workflow/workflow/model/Gateway.go 文件中）

我们看一下示例流程：



其中“天数判断”、“主管审批后同时分配给人事与副总”、“人事、副总全部同意后才能转到下一节点”等标黄节点，都是条件分支选择，在本项目中全都是依靠混合网关实现。

结束节点

结束节点在一个流程中可以有 N 个，其中任意一个到达，则流程结束。流程到达结束节点后，流程即标为结束，之后将流程数据做归档。

变量

变量在流程中起到重要作用,比如:

- 1、在设计流程时，各节点任务分配人、角色可以使用变量
- 2、分支选择时，使用表达式计算条件，表达式中可使用变量

流程变量可以在启动流程时加入，也可以在任务审核时加入/修改。

变量传入时，请在变量名前加入\$符号，如 \$day。

表达式

在上图示例流程中，存在一个分支判断：

A、如果请假天数<3 天，则直接通过并结束流程

B、如果请假天数>=3 天，则需要主管审批。

这里就用到了“条件表达式”。

请注意：作者偷懒，并没有自己做一个 ast 语义分析树，而是使用了 mysql 的 SELECT 表达式。比如流程定义中表达式为 $\$day > 3$ ，变量 \$day 为 3 的情况下，转换为 SQL 代码即为 $\text{SELECT } 3 > 3$ ，返回 0 (false)。

事件

事件根据触发时机分为几种：

- 1、节点开始时事件。在“流程转到节点”时触发。
- 2、节点结束时事件。在“节点完成,转入下一个节点”时触发。
- 3、任务完成时事件。任务完成(通过、驳回)时触发的事件，节点中可能产生 N 个任务，在每个任务完成时触发事件。
- 4、流程撤销事件。在流程被撤销时触发。

定时任务

定时任务可以实现“与时间有关的任务”。比如：如果任务超过 N 分钟未审核，则自动通过，或发送邮件通知。

流程实现示例

流程实现示例可参看 `easy-workflow/example` 下代码。其中：

`event` 文件夹下为示例事件；

`process` 文件夹下为流程定义；

`main.go` 文件中示例了流程引擎开启、打开引擎 WebAPI 的方法

三、主要 func 说明

1、流程处理方法在 github.com/Bunny3th/easy-workflow/workflow/engine 中定义

2、方法调用示例可以查看 `/easy-workflow/workflow/web_api/service` 包中代码

以下是主要方法说明

注：方法定义可能随版本变迁而有改变，以下仅为参考，实际请参照不同版本源码定义

流程定义相关

●流程 save/update

```
ProcessSave(Resource string, CreateUserID string) (int, error)
```

传入参数

Resource:JSON 格式的流程定义字符串

CreateUserID:创建人用户 ID

返回参数

流程 ID、error

●获取某个来源下所有的流程信息

```
GetProcessList(Source string) ([]database.ProcDef, error)
```

传入参数

Source:来源名

*注:流程引擎可能被多个系统共用。A 系统中有“请假流程”,B 系统中也可能有“请假流程”,

所以，流程都是有“来源”的。也可以理解为 package 名，每个流程是一个 func，需要归到一个 package 下面。

返回参数

流程定义数组、错误

•获取特定流程的定义

```
GetProcessDefine(ProcessID int) (Process, error)
```

传入参数：

ProcessID：流程 ID

返回参数：

流程定义 JSON、错误

流程实例相关

•流程开始

```
InstanceStart(ProcessID int, BusinessID string, Comment string, VariablesJson string) (int, error)
```

传入参数说明

ProcessID: 流程 ID

BusinessID: 工作流一般会与业务系统协同, 此处的 BusinessID, 即是业务系统中 ID。

Comment:批注

VariablesJson:变量, KV 对形式的 JSON 字符串。

示例: [{"Key":"starter","Value":"User001"}, {"Key":"days","Value":"5"}]

此示例中, 变量\$starter 值为" User001" ;变量\$days 值为 "5"

返回参数

流程实例 ID、错误

•流程实例撤销

```
InstanceRevoke(ProcessInstanceID int, Force bool, RevokeUserID string) error
```

传入参数

ProcessInstanceID: 流程实例 ID

Force: 通常情况下, 只有撤销人为流程发起人时才能撤销流程。此处如果设置

Force 为 true, 则可强制撤销, 不做用户检查

RevokeUserID: 撤销人 ID

返回参数

错误

●获取特定流程实例信息

```
GetInstanceInfo(ProcessInstanceID int) (Instance, error)
```

传入参数

ProcessInstanceID: 流程实例 ID

返回参数

流程实例信息、错误

●特定流程实例中审批记录

```
GetInstanceTaskHistory(ProcessInstanceID int) ([]Task, error)
```

传入参数

ProcessInstanceID: 流程实例 ID

返回参数

任务数组、错误

●获取 “由我发起” 的流程列表

```
GetInstanceStartByUser(UserID string, ProcessName string, StartIndex  
int, MaxRows int) ([]Instance, error)
```

传入参数

UserID: 用户 ID

ProcessName:指定流程名称,传入""则为全部

StartIndex:分页用,记录集起始 index

MaxRows:分页用,最大返回行数

返回参数

实例信息数组、错误

任务相关

●通过

```
TaskPass(TaskID int, Comment string, VariableJson string,  
DirectlyToWhoRejectedMe bool) error
```

传入参数

TaskID: 任务 ID

Comment: 批注

VariableJson: 变量, KV 对形式的 JSON 字符串。

示例: [{"Key":"starter","Value":"User001"}, {"Key":"days","Value":"5"}]

DirectlyToWhoRejectedMe: 任务通过后直接跳转到驳回我的节点。

返回参数

错误

●驳回

```
TaskReject(TaskID int, Comment string, VariableJson string) error
```

传入参数

TaskID: 任务 ID

Comment: 批注

VariableJson: 变量, KV 对形式的 JSON 字符串。

示例: [{"Key":"starter","Value":"User001"}, {"Key":"days","Value":"5"}]

返回参数

错误

●将任务转交给其他人

```
TaskTransfer(TaskID int, Users []string) error
```

传入参数

TaskID: 任务 ID

Users: 转交用户数组

返回参数

错误

●待办任务列表

```
GetTaskToDoList(userID string, ProcessName string, SortByASC bool,  
StartIndex int, MaxRows int) ([]Task, error)
```

传入参数说明

UserID:用户 ID

ProcessName:指定流程名称,传入""则为全部

SortByASC 返回数据是否按照任务生成时间升序排列(实际是按照 TaskID 排序。TaskID 是 int 型自增字段, 用其排序与用 createtime 效果一致)。若传入 false, 则会按照降序排列

StartIndex:分页用,返回数据集起始 index

MaxRows:分页用,最大返回行数

返回参数说明

任务信息数组、错误

●已办任务列表

```
GetTaskFinishedList(userID string, ProcessName string,  
IgnoreStartByMe bool, SortByASC bool, StartIndex int, MaxRows int)  
([]Task, error)
```

传入参数说明

获取特定用户已完成任务列表。参数说明：

UserID:用户 ID

ProcessName:指定流程名称,传入""则为全部

IgnoreStartByMe: 某些情况下只希望看到“别人提交由我审批完成的任务”,而不希望看到
“由我开启流程,而生成处理人是我自己的任务”,则传 True

SortByASC 返回数据是否按照任务完成时间升序排列。若传入 false, 则会按照降序排列

StartIndex:分页用,返回数据集起始 index

MaxRows:分页用,最大返回行数

返回参数说明

任务信息数组、错误

●获取特定任务信息

```
GetTaskInfo(TaskID int) (Task, error)
```

传入参数说明

TaskID:任务 ID

返回参数说明

任务信息、错误

●获取本任务所在节点的所有上游节点

注：此方法为搭配自由驳回功能使用

```
TaskUpstreamNodeList(TaskID int) ([]Node, error)
```

传入参数说明

TaskID:任务 ID

返回参数说明

节点数组、错误

●自由驳回

注：任务驳回到任意一个上游节点，会签节点不可使用此功能

```
TaskFreeRejectToUpstreamNode(TaskID int, NodeID string, Comment  
string, VariableJson string) error
```

传入参数说明

TaskID:任务 ID

NodeID：需要驳回到节点 ID

Comment：批注

VariableJson：变量，KV 对形式的 JSON 字符串。

示例：[{"Key":"starter","Value":"User001"}, {"Key":"days","Value":"5"}]

返回参数说明

错误

●列出当前任务可以执行哪些操作

除了传统的通过驳回，本项目还增加了"自由驳回"与"直接提交到上一个驳回我的节点"

而"直接提交到上一个驳回我的节点"：

- 1、在会签节点无法使用
- 2、在此任务的上一节点并未做驳回时也无法使用

对于前端而言，实现无法提前知道这些信息。

难道让用户一个一个点按钮试错？此方法方便前端判断，某一个任务可以执行哪些操作

```
WhatCanIDo(TaskID int) (TaskAction, error)
```

传入参数说明

TaskID：任务 ID

返回参数说明

可执行操作信息、错误

四、数据库表结构说明

注：目前项目中共有 10 张表，其中历史归档表前缀为 hist,表结构与在线表一致，故此处仅介绍 5 张在线表。

流程定义资源表

proc_def

	Field	Type	Comment
🔑	id	int unsigned NOT NULL	流程ID
	name	varchar(250) NOT NULL	流程名字
	version	int unsigned NOT NULL	版本号
	resource	text NOT NULL	流程定义模板
	user_id	varchar(250) NOT NULL	创建者ID
	source	varchar(250) NOT NULL	来源(引擎可能被多个系统、组件等使用，这里记下从哪个来源创建的流程)
	create_time	datetime NULL	创建时间


流程节点关系表

proc_execution

	Field	Type	Comment
🔑	id	int unsigned NOT NULL	
	proc_id	int unsigned NOT NULL	流程ID
	proc_version	int unsigned NOT NULL	流程版本号
	node_id	varchar(250) NOT NULL	节点ID
	node_name	varchar(250) NOT NULL	节点名称
	prev_node_id	varchar(250) NULL	上级节点ID
	node_type	tinyint NOT NULL	流程类型 0:开始节点 1:任务节点 2:网关节点 3:结束节点
	is_cosigned	tinyint NOT NULL	是否会签
	create_time	datetime NULL	创建时间


流程实例表

proc_inst

	Field	Type	Comment
	id	int unsigned NOT NULL	流程实例ID
	proc_id	int NOT NULL	流程ID
	proc_version	int unsigned NOT NULL	流程版本号
	business_id	varchar(250) NULL	业务ID
	starter	varchar(250) NOT NULL	流程发起人用户ID
	current_node_id	varchar(250) NOT NULL	当前进行节点ID
	create_time	datetime NULL	创建时间
	status	tinyint NULL	0:未完成(审批中) 1:已完成(通过) 2:撤销

流程实例中变量定义表

proc_inst_variable

	Field	Type	Comment
	id	int unsigned NOT NULL	
	proc_inst_id	int unsigned NOT NULL	流程实例ID
	key	varchar(250) NOT NULL	变量key
	value	varchar(250) NOT NULL	变量value

流程任务表

proc_task

Field	Type	Comment
 id	int unsigned NOT NULL	任务ID
proc_id	int unsigned NOT NULL	流程ID,冗余字段,偷懒用
proc_inst_id	int unsigned NOT NULL	流程实例ID
business_id	varchar(250) NULL	业务ID,冗余字段,偷懒用
starter	varchar(250) NOT NULL	流程发起人用户ID,冗余字段,偷懒用
node_id	varchar(250) NOT NULL	节点ID
node_name	varchar(250) NOT NULL	节点名称
prev_node_id	varchar(250) NULL	上个处理节点ID,注意这里和execution中的上一个节点不一样,这里是实际审批处理时上个已处理节点的ID
is_cosigned	tinyint NULL	0:任意一人通过即可 1:会签
batch_code	varchar(50) NULL	批次码.节点会被驳回,一个节点可能产生多批task,用此码做分别
user_id	varchar(250) NOT NULL	分配用户ID
status	tinyint NULL	任务状态:0:初始 1:通过 2:驳回
is_finished	tinyint NULL	0:任务未完成 1:处理完成.任务未必都是用户处理的,比如会签时一人驳回,其他任务系统自动设为已处理
comment	text NULL	任务备注
proc_inst_create_time	datetime NOT NULL	流程实例创建时间,冗余字段,偷懒用
create_time	datetime NULL	系统创建任务时间
finished_time	datetime NULL	处理任务时间

五、源码文件结构说明

路径	说明
github.com/Bunny3th/easy-workflow/example	示例代码
.../docs	Swagger 描述文件
.../event	流程事件示例
.../process	流程定义示例
main.go	流程开启示例
github.com/Bunny3th/easy-workflow/workflow	流程核心代码
.../database	数据库表结构定义
.../engine	流程引擎代码
.../model	流程数据结构定义
.../web_api	使用 gin 引擎的 webAPI