工作流

> 采用Flowable开源工作流引擎，6.4.2版本。

概述

工作流能解决的问题

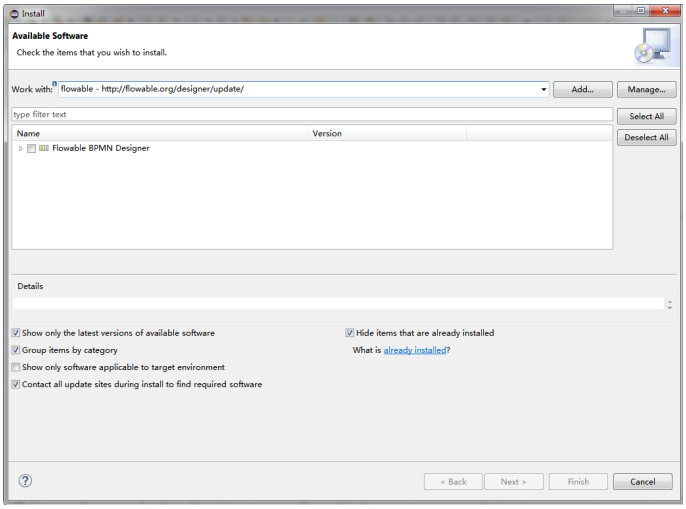
- 复杂的审批流程，图形化
- 审批流程中的事件流转等流程控制的代码从业务代码中剥离出来

工作流与业务代码结合的问题

- 工作流中的节点与我们系统中的角色对应关系。
- 工作流中消息通知。
- 工作流中数据传递等。

开发手册

安装流程图绘制插件



地址：http://flowable.org/designer/update/

添加依赖

```
<!-- flowable -->
                <dependency>
                        <groupId>org.flowable</groupId>
                        <artifactId>flowable-spring-boot-starter<
/artifactId>
                        <version>6.4.2</version>
                </dependency>
                <dependency>
                        <groupId>xerces</groupId>
                        <artifactId>xercesImpl</artifactId>
                        <version>2.12.0</version>
                </dependency>
                <!-- flowable -->
```

代码样例

```
@Autowired
    private RuntimeService runtimeService;

    @Autowired
    private TaskService taskService;

    @Autowired
    private ProcessEngine processEngine;

    @Autowired
    private RepositoryService repositoryService;

    /**
     * . taskProcessInstanceId.
     *
     * @param userId
     * @param businessId
     * @return Task
     */
    public Task createTask(String userId, String businessId) {
        //
        Map<String, Object> map = new HashMap<>();
        map.put("taskUser", userId);
        map.put("businessId", businessId);
        ProcessInstance processInstance = runtimeService.
startProcessInstanceByKey("mas", map);
        Task task = taskService.createTaskQuery().processInstanceId
(processInstance.getId())
                .singleResult();

        //
        return approve(task.getProcessInstanceId());
```

```
    }

    /**
     * process idtask.
     *
     * @param processInstanceId
     * @return
     */
    public Task getCurrentTask(String processInstanceId) {
        return taskService.createTaskQuery().processInstanceId
(processInstanceId).singleResult();
    }

    /**
     * .
     *
     * @param taskName taskName.
     * @return List<Task>
     */
    public List<Task> queryUnAssigneeTask(String taskName) {
        return taskService.createTaskQuery().taskName(taskName).
orderByTaskAssignee().asc().list()
                .stream().filter(t -> t.getAssignee() == null).collect
(Collectors.toList());
    }

    /**
     * .
     *
     * @param taskId Id
     * @param userId id
     */
    public void accept(String processInstanceId, String userId) {
        Task task = getCurrentTask(processInstanceId);
        taskService.claim(task.getId(), userId);
    }

    /**
     * .
     *
     * @param processInstanceId processInstanceId
     * @return Task
     */
    public Task approve(String processInstanceId) {
        Task task = getCurrentTask(processInstanceId);
        Map<String, Object> variables = new HashMap<>();
        variables.put("message", "");
        taskService.complete(task.getId(), variables);
        return taskService.createTaskQuery().processInstanceId
(processInstanceId).singleResult();
    }

    /**
```

```
     * .
     *
     * @param processInstanceId processInstanceId
     * @return Task
     */
    public Task reject(String processInstanceId) {
        Task task = getCurrentTask(processInstanceId);
        Map<String, Object> variables = new HashMap<>();
        variables.put("message", "");
        taskService.complete(task.getId(), variables);
        return taskService.createTaskQuery().processInstanceId
(processInstanceId).singleResult();
    }


    /**
     *
     *
     * @param processId processId
     * @return InputStream
     */
    public InputStream genProcessDiagram(String processId) {
        ProcessInstance pi = runtimeService.createProcessInstanceQuery()
                .processInstanceId(processId).singleResult();
        //
        if (pi == null) {
            return null;
        }
        Task task = taskService.createTaskQuery().processInstanceId(pi.
getId()).singleResult();
        // ID
        String InstanceId = task.getProcessInstanceId();
        List<Execution> executions = runtimeService.createExecutionQuery()
                .processInstanceId(InstanceId).list();

        // ActivityId
        List<String> activityIds = new ArrayList<>();
        List<String> flows = new ArrayList<>();
        for (Execution exe : executions) {
            List<String> ids = runtimeService.getActiveActivityIds(exe.
getId());
            activityIds.addAll(ids);
        }

        //
        BpmnModel bpmnModel = repositoryService.getBpmnModel(pi.
getProcessDefinitionId());
        ProcessEngineConfiguration engconf = processEngine.
getProcessEngineConfiguration();
        ProcessDiagramGenerator diagramGenerator = engconf.
getProcessDiagramGenerator();
        return diagramGenerator.generateDiagram(bpmnModel, "png",
activityIds, flows,
```

```
                engconf.getActivityFontName(), engconf.getLabelFontName(),
                engconf.getAnnotationFontName(), engconf.getClassLoader(),
1.0, true);

    }
```