

Activiti workflow environment building

[课前默写](#)

[课程回顾](#)

[今日内容](#)

[教学目标](#)

[第一章 工作流简述](#)

[1.1、工作流是什么](#)

[1.2、哪些行业需要工作流](#)

[1.3、工作流系统的使用](#)

[第二章 Activiti的历史简介](#)

[第三章 重要名词解释](#)

[3.1、BPM](#)

[3.2、BPMN](#)

[第四章 环境搭建](#)

[4.1 下载源码包](#)

[4.2 通过java代码创建表](#)

[4.3 idea安装activiti插件](#)

[4.4 示例工程](#)

[4.5 Activiti所支持数据库类型](#)

[第五章 Activiti整体结构图](#)

[5.1、Service服务说明](#)

[5.1.1、RepositoryService仓库服务](#)

[5.1.2、运行时服务](#)

[5.1.3、任务服务](#)

[5.1.4、认证服务](#)

[5.1.5、历史服务](#)

[5.1.6、历史服务](#)

[总结](#)

[作业](#)

[面试题](#)

课前默写

1. 写出Spring Data JPA中的注解
2. 写出JPQL、HQL、SQL区别
3. 写出Spring Data JPA的查询方式

课程回顾

1. SpringData JPA
2. SpringData JPA使用
3. JPQL语法
4. 查询方式
5. SSS框架整合

今日内容

1. 工作流的基本理论
2. Activiti简介及BPMN
3. 使用IDEA搭建Activiti开发环境
4. Activiti的整体结构
5. Process流程定义和部署维护

教学目标

1. 掌握工作流的基本理论
2. 掌握Activiti简介及BPMN
3. 掌握使用IDEA搭建Activiti开发环境
4. 掌握Activiti的整体结构
5. 掌握Process流程定义和部署维护

第一章 工作流简述

1.1、工作流是什么

工作流(Workflow)，就是通过计算机对业务流程自动化执行管理。它主要解决的是“使在多个参与者之间按照某种预定义的规则自动进行传递文档、信息或任务的过程，从而实现某个预期的业务目标，或者促使此目标的实现”。

工作流就是通过计算机技术对业务流程进行自动化管理。实现多个参与者按照预定的流程去自动执行业务流程。

1.2、哪些行业需要工作流

消费品行业，制造业，电信服务业，银证险等金融服务业，物流服务业，物业服务业，物业管理，大中型进出口贸易公司，政府事业机构，研究院所及教育服务业等，特别是大的跨国企业和集团公司。

总结一句话：凡是有组织机构的公司都有可能用到工作流。

1.3、工作流系统的使用

1. 关键业务流程：订单、报价处理、采购处理、合同审核、客户电话处理、供应链管理等
2. 行政管理类：出差申请、加班申请、请假申请、用车申请、各种办公用品申请、购买申请、日报周报等凡是原来手工流转处理的行政表单。
3. 人事管理类：员工培训安排、绩效考评、职位变动处理、员工档案信息管理等。
4. 财务相关类：付款请求、应收款处理、日常报销处理、出差报销、预算和计划申请等。
1. 客户服务类：客户信息管理、客户投诉、请求处理、售后服务管理等。
2. 特殊服务类：ISO系列对应流程、质量管理对应流程、产品数据信息管理、贸易公司报关处理、

物流公司货物跟踪处理等各种通过表单逐步手工流转完成的任务均可应用工作流软件自动规范地实施。

第二章 Activiti的历史简介

Activiti项目是一项新的基于Apache许可的开源BPM平台，从基础开始构建，旨在提供支持新的BPMN 2.0标准，包括支持对象管理组（OMG），面对新技术的机遇，诸如互操作性和云架构，提供技术实现。

创始人Tom Baeyens是JBoss jBPM的项目架构师，以及另一位架构师Joram Barrez，一起加入到创建Alfresco这项首次实现Apache开源许可的BPMN 2.0引擎开发中来。

Activiti前身是jbpm4，Activiti架构和jbpm4基本上是一样的。

架构师Tom Baeyens说：“Activiti有非常大的影响力来改变目前BPM的生态。Activiti的Apache授权，完整的功能，将使Activiti到达一个新的水平。Activiti将推动业界的创新，因为BPM技术可以广泛而自由地被应用。通过实现这些想法以及开源社区的努力，也让Activiti成为事实上的BPM和BPMN标准执行”。

第三章 重要名词解释

3.1、BPM

BPM，即业务流程管理，是一种以规范化的构造端到端的卓越业务流程为中心，以持续的提高组织业务绩效为目的的系统化方法，常见商业管理教育如EMBA、MBA等都将BPM包含在内。

3.2、BPMN

BPMN：业务流程建模与标注，包括这些图元如何组合成一个业务流程图（Business Process Diagram）；讨论BPMN的各种的用途，包括以何种精度来影响一个流程图中的模型；BPMN作为一个标准的价值，以及BPMN未来发展的远景。

业务流程图由一系列的图形化元素组成。这些元素简化了模型的开发，且业务分析者看上去非常熟悉。这些元素每个都有各自的特性，且与大多数的建模器类似。比如，活动是矩形，条件是菱形。应该强调的是：开发BPMN的动力就是为了在创建业务流程模型时提供一个简单的机制，同时又能够处理来自业务流程的复杂性。要处理这两个矛盾的需求的方法就是将标记的图形化方面组织分类为特定的类别。这里提供标记类别中的一小部分，以便业务流程图的读者可以简单地识别出元素的基本类型从而理解图形。以下是四种基本的类型：

- 1) 流对象(Flow)
- 2) 连接对象(Connection)
- 3) 泳道(Swimlane)
- 4) 人工信息(Artifact)

流对象：

一个业务流程图有三个流对象的核心元素。

这三种流对象是

事件---一个事件用圆圈来描述，表示一个业务流程期间发生的事情。事件影响流程的流动，一般有一个原因（触发器）或一个影响（结果）。基于它们对流程的影响，有三种事件：开始，中间以及终止事件

活动---一个活动用圆角矩形表示，是要处理工作的一般术语。一个活动可以是原子性的也可以是非原子性的（可以由多个活动组合而成的更大粒度的活动）。活动的类型包括：任务和子流程。子流程在图形的下方中间外加一个小加号（+）来区分。

条件---条件用熟悉的菱形表示，用于控制序列流的分支与合并。另外，它还可以作为传统的选择，还包括路径的分支与合并。其内部的标记会给出控制流的类型。

连接对象：

连接对象将流对象连接起来形成一个业务流程的基本结构。提供此功能的三个连接对象是：

顺序流---顺序流用一个带实心箭头的实心线表示，用于指定活动执行的顺序。注意“控制流”这个术语一般不用于BPMN

消息流---消息流用一条带有开箭头的虚线表示，用于描述两个独立的业务参与者（业务实体或业务角色）之间发送和接受的消息流动。在BPMN中，用两个独立的池代表两个参与者。

关联---用一根带有线箭头的点线表示关联，用于将相关的数据、文本和其他人工信息与流对象联系起来。关联用于展示活动的输入和输出。

泳道：

许多建模技术利用泳道这个概念将活动划分到不同的可视化类别中来描述由不同的参与者的责任与职责。BPMN支持2种主要的泳道构件。

池---池描述流程中的一个参与者。可以看做是将一系列活动区别于其他池的一个图形容器，一般用于B2B的上下文中。

道---道就是在池里面再细分，可以是垂直的也可以是水平的。道也是用于组织和分类活动。

人工信息：

人工信息添加到建模的业务流程上下文中作为信息备注，便于人员理解，当前BPMN规范的版本预定义了3种人工信息：

数据对象---数据对象是一个显示活动是如何需要或产生数据的。它们通过关联与活动连接起来。

组---组用一个虚线的圆角矩形表示，用于记录或分析的目的，但不影响顺序流。

第四章 环境搭建

开发环境：

activiti 5.22.0

jdk 1.8

mysql 5.6

tomcat 8.5

注意，activiti需要在数据库里创建表，可以在官网下载源码安装包，里面有数据库的表。或者通过java代码自动创建表。所以1.1和1.2节中的方法二选一即可。








4.1 下载源码包

<https://www.activiti.org/download-links>




Previous Releases

| | | | |
|-------------------------------------|--------|----------------|-------------|
| activiti-6.0.0.zip | 115 MB | Apache License | 26 May 2017 |
| activiti-5.22.0.zip | 90 MB | Apache License | 9 Dec 2016 |


























本教程使用的是5.22.0版本，解压后如下：

| | | | |
|---|------------------|-------------------|--------|
|  database | 2017/12/10 19:58 | 文件夹 | |
|  docs | 2017/12/10 19:58 | 文件夹 | |
|  libs | 2017/12/10 19:58 | 文件夹 | |
|  wars | 2018/3/30 16:49 | 文件夹 | |
|  license.txt | 2016/11/3 15:12 | 文本文档 | 12 KB |
|  notice.txt | 2016/11/3 15:12 | 文本文档 | 9 KB |
|  readme.html | 2016/11/3 15:12 | Firefox HTML D... | 228 KB |

在database\create文件夹下有各种版本数据库使用的创建数据库的sql语句。5.22.0版本里有25张基础表。此处使用mysql5.6数据库，新建一个数据库，运行如下文件

| | | | |
|--|-----------------|--------|-------|
|  activiti.mysql.create.engine.sql | 2016/11/3 15:12 | SQL 文件 | 10 KB |
|  activiti.mysql.create.history.sql | 2016/11/3 15:12 | SQL 文件 | 6 KB |
|  activiti.mysql.create.identity.sql | 2016/11/3 15:12 | SQL 文件 | 2 KB |

运行后得到25张表：

| | |
|---|---------------------|
|  | act_evt_log |
|  | act_ge_bytearray |
|  | act_ge_property |
|  | act_hi_actinst |
|  | act_hi_attachment |
|  | act_hi_comment |
|  | act_hi_detail |
|  | act_hi_identitylink |
|  | act_hi_procinst |
|  | act_hi_taskinst |
|  | act_hi_varinst |
|  | act_id_group |
|  | act_id_info |
|  | act_id_membership |
|  | act_id_user |
|  | act_procdef_info |
|  | act_re_deployment |
|  | act_re_model |
|  | act_re_procdef |
|  | act_ru_event_subscr |
|  | act_ru_execution |
|  | act_ru_identitylink |
|  | act_ru_job |
|  | act_ru_task |
|  | act_ru_variable |

数据库表说明：

| 简介 | | |
|----|---------|--|
| # | 前缀 | 描述 |
| 1 | ACT_RE_ | RE表示Repository资源库，保存流程定义，模型等设计阶段的数据。 |
| 2 | ACT_RU_ | RU表示Runtime运行时，保存流程实例，任务，变量等运行阶段的数据。 |
| 3 | ACT_HI_ | HI表示History历史，保存历史实例，历史任务等流程历史数据。 |
| 4 | ACT_ID_ | ID表示Identity身份，保存用户，群组，关系等组织机构相关数据。（Activiti中的组织机构过于简单，仅用于演示。） |
| 5 | ACT_GE_ | GE表示General通用，属于一些通用配置。 |
| 6 | 其他 | ACT_EVT_LOG和ACT_PROCDEF_INFO没有按照规则来，两者分别属于HI和RE。 |

4.2 通过java代码创建表

引入依赖的jar

```
<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>
```

```
<version>4.12</version>

</dependency>

<dependency>

  <groupId>org.activiti</groupId>

  <artifactId>activiti-bpmn-converter</artifactId>

  <version>5.22.0</version>

</dependency>

<dependency>

  <groupId>org.activiti</groupId>

  <artifactId>activiti-bpmn-model</artifactId>

  <version>5.22.0</version>

</dependency>

<dependency>

  <groupId>org.activiti</groupId>

  <artifactId>activiti-image-generator</artifactId>

  <version>5.22.0</version>

</dependency>

<dependency>

  <groupId>org.activiti</groupId>

  <artifactId>activiti-process-validation</artifactId>

  <version>5.22.0</version>

</dependency>

<dependency>

  <groupId>org.slf4j</groupId>
```

```
<artifactId>slf4j-log4j12</artifactId>

<version>1.6.1</version>

</dependency>

<dependency>

  <groupId>org.slf4j</groupId>

  <artifactId>slf4j-api</artifactId>

  <version>1.6.1</version>

</dependency>

<dependency>

  <groupId>mysql</groupId>

  <artifactId>mysql-connector-java</artifactId>

  <version>5.1.44</version>

</dependency>

<dependency>

  <groupId>commons-dbcp</groupId>

  <artifactId>commons-dbcp</artifactId>

  <version>1.4</version>

</dependency>

<dependency>

  <groupId>org.activiti</groupId>

  <artifactId>activiti-root</artifactId>

  <version>5.22.0</version>

</dependency>

<dependency>

  <groupId>org.activiti</groupId>
```



```

        <artifactId>activiti-spring</artifactId>

        <version>5.22.0</version>

    </dependency>

</dependencies>

```

在resources目录下创建activiti.cfg.xml，配置数据源和默认引擎

activiti.cfg.xml

```

<beans xmlns="http://www.springframework.org/schema/beans"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd">
    <bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource" >
        <property name="driverClassName" value="com.mysql.jdbc.Driver" />
        <property name="url" value="jdbc:mysql://localhost:3306/db_activiti?
characterEncoding=utf-8" />
        <property name="username" value="root" />
        <property name="password" value="xph666" />
    </bean>
    <bean id="processEngineConfiguration"
        class="org.activiti.engine.impl.cfg.StandaloneProcessEngineConfiguration">
        <property name="dataSource" ref="dataSource" />
        <property name="databaseSchemaUpdate" value="true" />
    </bean>
</beans>

```

其中databaseSchemaUpdate属性允许在进程引擎启动和关闭时设置策略来处理数据库模式。

- 1) false（默认）：当创建流程引擎时，检查数据库模式对库的版本，如果版本不匹配则抛出异常。
- 2) true：在构建流程引擎时，执行检查，如果需要，执行模式的更新。如果模式不存在，则创建它。
- 3) create-drop：在创建流程引擎时创建模式，并在流程引擎关闭时删除模式。

这里设置成true，运行程序的时候自动生成表。

activiti底层使用的是mybatis，可以通过log4j查看执行的语句：

log4j.properties

```
# For all other servers: Comment out the Log4J listener in web.xml to
activate Log4J.

log4j.rootLogger=DEBUG, stdout

log4j.appender.stdout=org.apache.log4j.ConsoleAppender

log4j.appender.stdout.layout=org.apache.log4j.PatternLayout

log4j.appender.stdout.layout.ConversionPattern=%d %p [%c] - %m%n

#begin

#for normal test,delete when online

log4j.logger.com.ibatis=DEBUG

log4j.logger.com.ibatis.common.jdbc.SimpleDataSource=DEBUG

log4j.logger.com.ibatis.common.jdbc.ScriptRunner=DEBUG

log4j.logger.com.ibatis.sqlmap.engine.impl.SqlMapClientDelegate=DEBUG

log4j.logger.java.sql.Connection=DEBUG

log4j.logger.java.sql.Statement=DEBUG

log4j.logger.java.sql.PreparedStatement=DEBUG

log4j.logger.java.sql.ResultSet=DEBUG
```

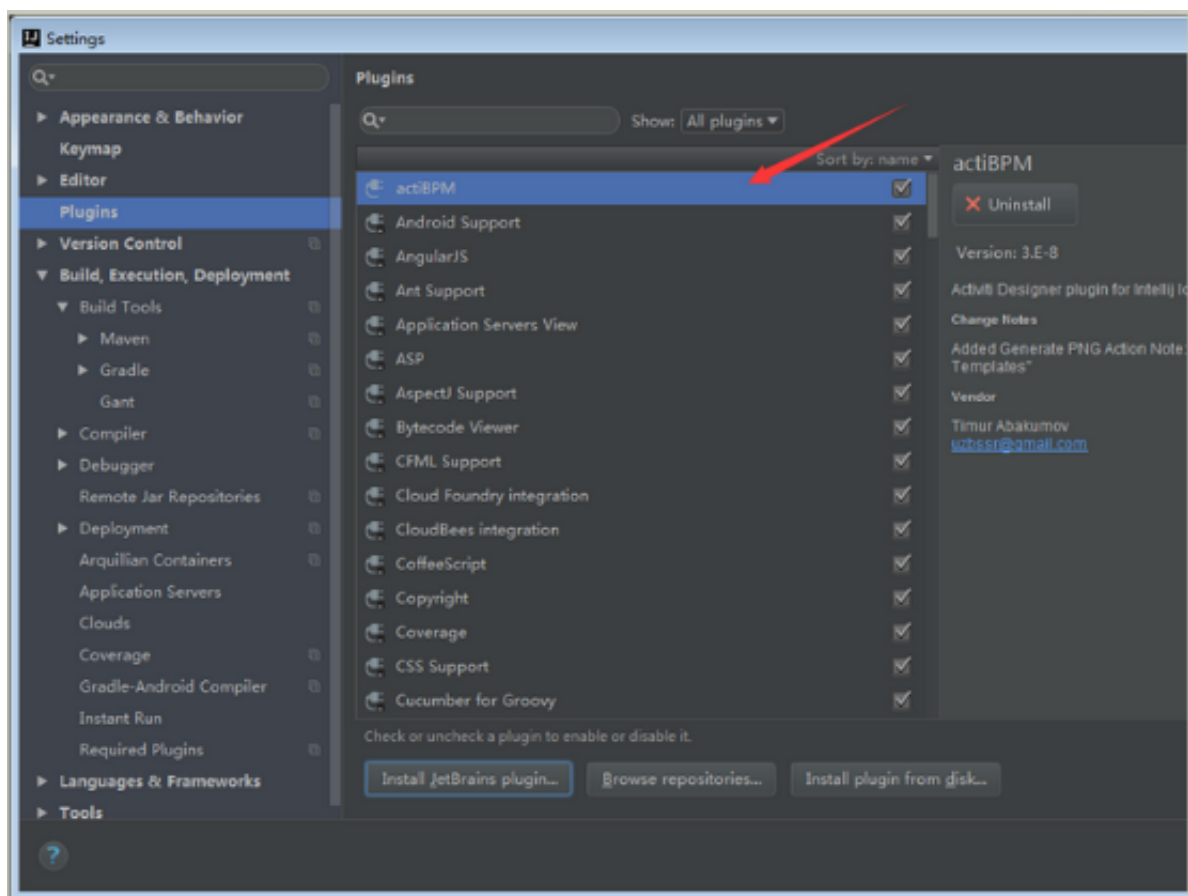
测试代码：

```
@Test
public void createTable() {
    ProcessEngineConfiguration configuration =
        ProcessEngineConfiguration.createProcessEngineConfigurationFromResource("activiti.cfg.xml");
    ProcessEngine processEngine = configuration.buildProcessEngine();
    System.out.println(processEngine);
}
```

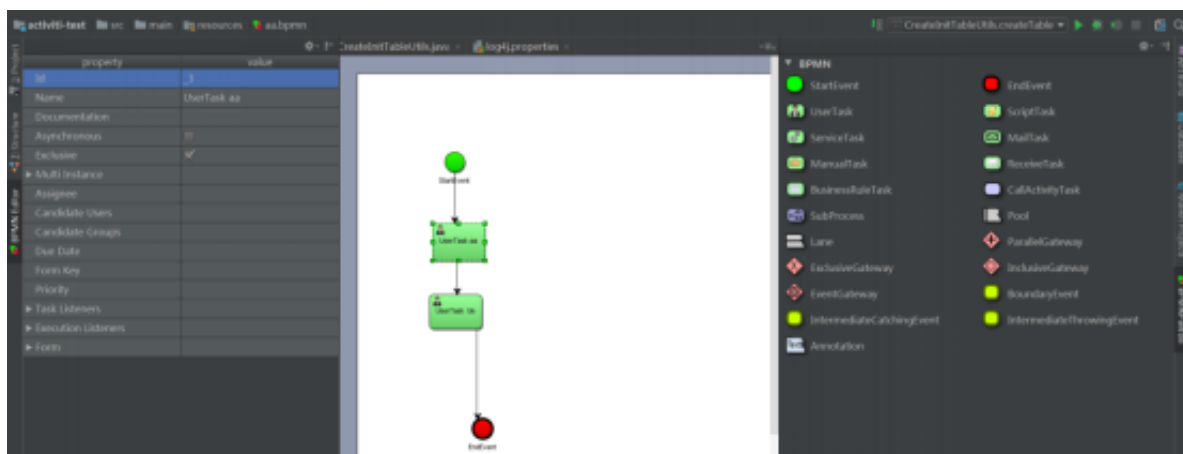
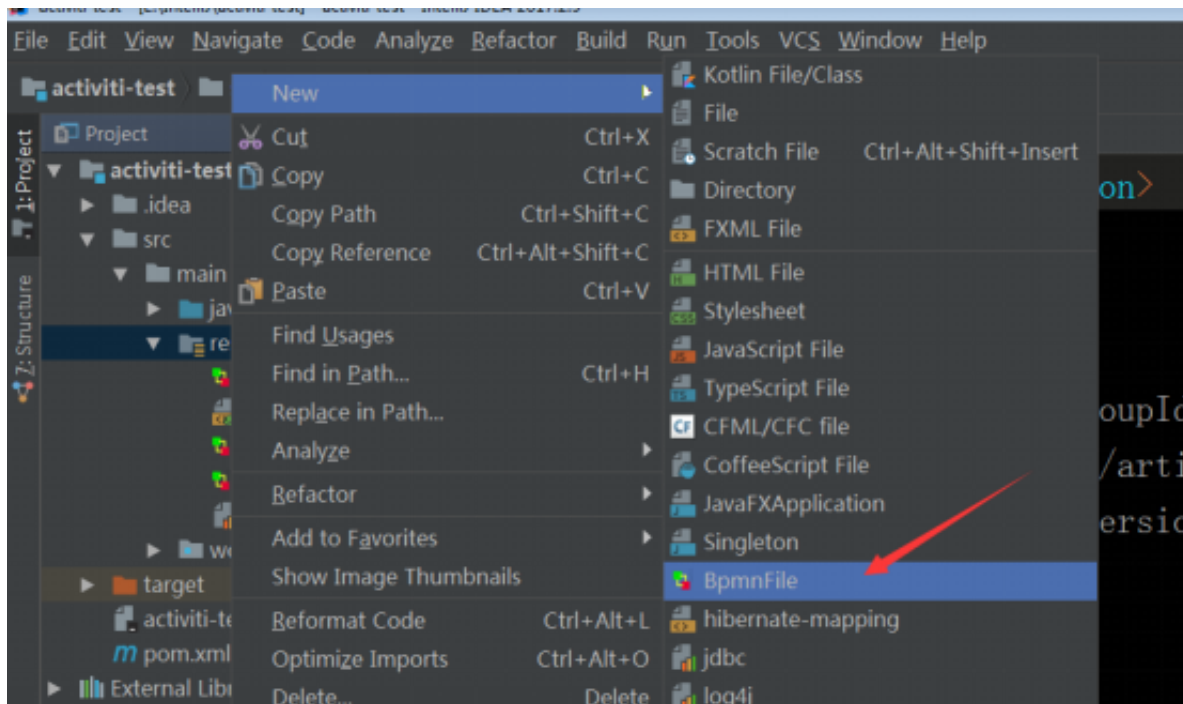
4.3 idea安装activiti插件

File->Settings->Plugins

搜索actiBPM，右侧选择Install，安装成功后重启Idea。



在工程中可以创建 bpmn文件了。bpmn规范会在后续课程中讲解。



idea中绘制流程图的时候中文会乱码，在idea安装路径的bin目录下修改配置文件：

| | | |
|----------------------|------------------|------------|
| idea.exe.vmoptions | 2017/9/29 14:55 | VMOPTIONS |
| idea.ico | 2017/9/29 14:55 | 图标 |
| idea.properties | 2017/11/28 11:27 | PROPERTIES |
| idea.properties.bak | 2017/11/28 11:27 | BAK 文件 |
| idea64.exe | 2017/9/29 14:55 | 应用程序 |
| idea64.exe.vmoptions | 2017/9/29 14:55 | VMOPTIONS |

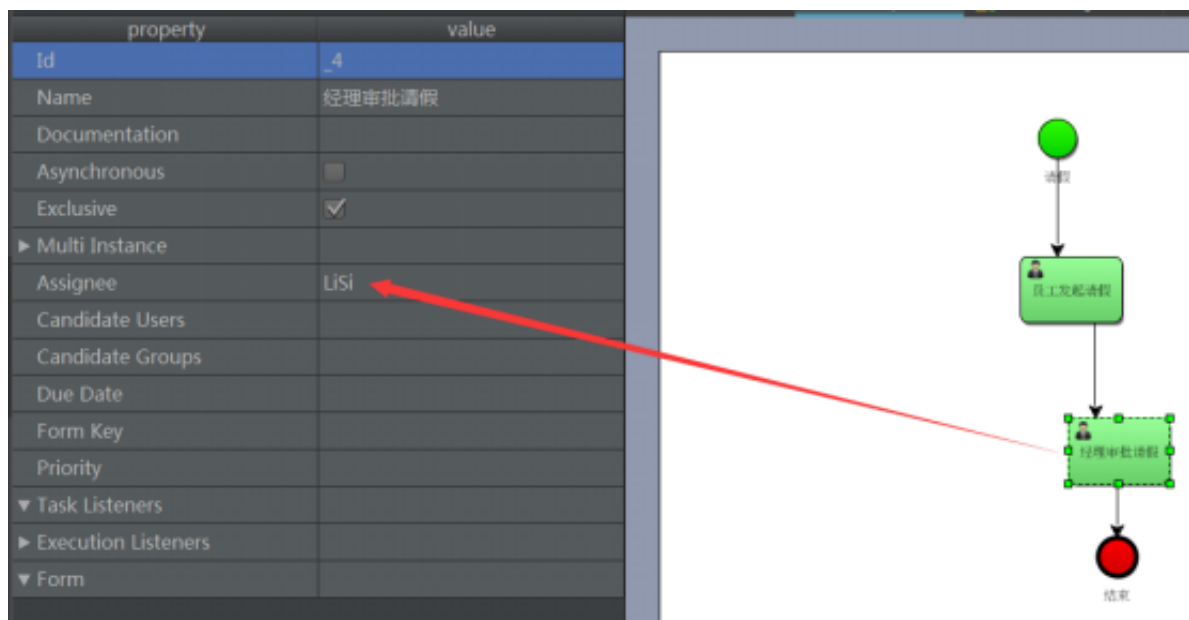
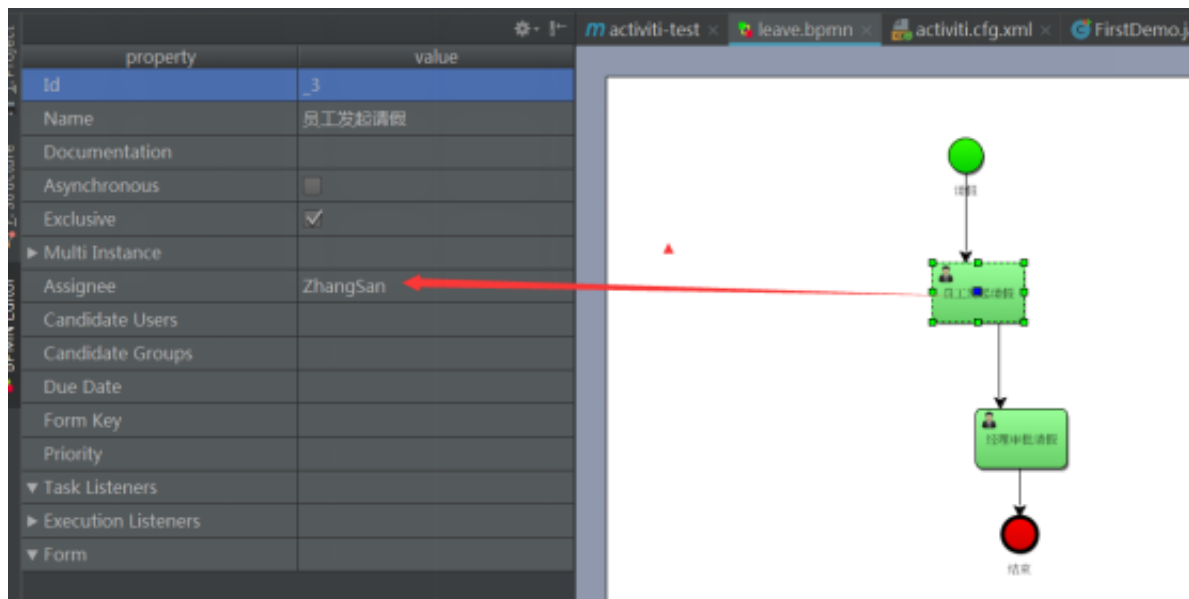
在文件末尾加一句，如图：-Dfile.encoding=UTF-8

```

1 -Xms128m
2 -Xmx750m
3 -XX:ReservedCodeCacheSize=240m
4 -XX:+UseConcMarkSweepGC
5 -XX:SoftRefLRUPolicyMSPerMB=50
6 -ea
7 -Dsun.io.useCanonCaches=false
8 -Djava.net.preferIPv4Stack=true
9 -XX:+HeapDumpOnOutOfMemoryError
10 -XX:-OmitStackTraceInFastThrow
11 -Dfile.encoding=UTF-8
  
```

4.4 示例工程

绘制流程图leave.bpmn: ZhangSan发起请假, LiSi审批请假



部署任务:

```
@Test
public void addDeployment() {
    Deployment deployment = processEngine.getRepositoryService() // 与流程定义和部署对象相关的Service.createDeployment() // 创建一个部署对象
    .name("请假流程") // 设置对应流程的名称
    .addClasspathResource("leave.bpmn")
    .deploy(); // 完成部署
}
```

数据库里已经插入了对应的数据

| ID_ | NAME_ | CATEGORY_ | TENANT_ID_ | DEPLOY_TIME_ |
|------|-------|-----------|------------|-------------------------|
| 2501 | 请假流程 | (Null) | | 2018-04-02 15:15:19.259 |

| ID_ | REV_ | NAME_ | DEPLOYMENT_ID_ | BYTES_ | GENERATED_ |
|------|------|-----------------------|----------------|--------|------------|
| 2502 | 1 | leave.bpmn | 2501 | (BLOB) | 0 |
| 2503 | 1 | leave.myProcess_1.png | 2501 | (BLOB) | 1 |

| ID_ | REV_ | CATEGORY_ | NAME_ | KEY_ | VERSION_ | DEPLOYMENT_ID_ | RESOURCE_NAME_ | DGRM_RESOURCE_NAME_ |
|------------------|------|-----------------------------|--------|-------------|----------|----------------|----------------|-----------------------|
| myProcess_1:2504 | 1 | http://www.activiti.org/tes | (Null) | myProcess_1 | 1 | 2501 | leave.bpmn | leave.myProcess_1.png |

启动任务：

```
@Test
public void startProcess() {
    RuntimeService runtimeService = processEngine.getRuntimeService();
    //key是act_re_procdef中的KEY_,bpmn的id
    runtimeService.startProcessInstanceByKey("myProcess_1");
}
```

查看ZhangSan的任务并提交给下一个操作者。查看李四的任务并处理。(act_ru_task表)

```
@Test
public void queryZhangSanTask() {
    String assignee = "ZhangSan";
    List<Task> taskList = processEngine.getTaskService().getTaskList(assignee);
    service.createTaskQuery().taskAssignee(assignee).list();

    for (Task task : taskList) {
        System.out.println("代办任务ID:" + task.getId());
        System.out.println("代办任务name:" + task.getName());

        System.out.println("代办任务创建时间:" + task.getCreateTime());

        System.out.println("代办任务办理人:" + task.getAssignee());

        System.out.println("流程实例ID:" + task.getProcessInstanceId());
    }
}
```

```

        System.out.println("执行对象ID:" + task.getExecutionId**());

        //提交任务到下一个代理人
        engine.getTaskService().complete(task.getId());
    }
}

@Test
public void queryLiSiTask() {
    String assignee = "LiSi";

    List<Task> taskList = processEngine.getTaskService()//获取任务service
        .createTaskQuery()//创建查询对象
        .taskAssignee(assignee)//指定查询人
        .list();

    for (Task task : taskList) {
        System.out.println("代办任务ID:" + task.getId());

        System.out.println("代办任务name:" + task.getName());

        System.out.println("代办任务创建时间:" + task.getCreateTime**());

        System.out.println("代办任务办理人:" + task.getAssignee**());

        System.out.println("流程实例ID:" + task.getProcessInstanceId());

        System.out.println("执行对象ID:" + task.getExecutionId());

        engine.getTaskService().complete(task.getId());

    }
}

```

查看执行历史act_hi_taskinst表:

```

@Test
public void viewHistory() {
    HistoryService historyService = processEngine.getHistoryService();
    HistoricProcessInstance historicProcessInstance = historyService

.createHistoricProcessInstanceQuery().processInstanceId("15001").singleResult();
    System.out.println("开始时间: " +
historicProcessInstance.getStartTime());
    System.out.println("结束时间: " + historicProcessInstance.getEndTime());
}

```

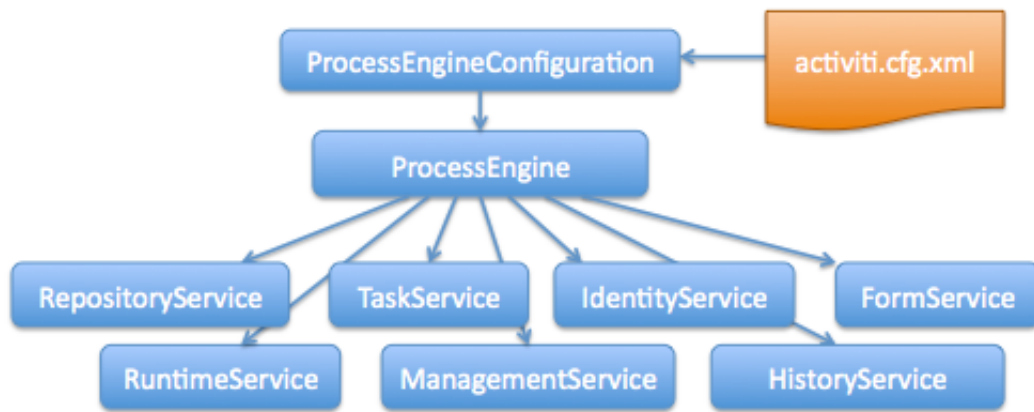
4.5 Activiti所支持数据库类型

| Activiti database type | Example JDBC URL | Notes |
|------------------------|---|---|
| h2 | jdbc:h2:tcp://localhost/activiti | Default configured database |
| mysql | jdbc:mysql://localhost:3306/activiti?autoReconnect=true | Tested using mysql-connector-java database driver |
| oracle | jdbc:oracle:thin:@localhost:1521:xe | |
| postgres | jdbc:postgresql://localhost:5432/activiti | |
| db2 | jdbc:db2://localhost:50000/activiti | |
| mssql | jdbc:sqlserver://localhost:1433;databaseName=activiti (jdbc.driver=com.microsoft.sqlserver.jdbc.SQLServerDriver) OR jdbc:jtds:sqlserver://localhost:1433/activiti (jdbc.driver=net.sourceforge.jtds.jdbc.Driver) | Tested using Microsoft JDBC Driver 4.0 (sqljdbc4.jar) and JTDS Driver |

第五章 Activiti整体结构图

- 1、流程引擎是整个activiti的核心，所有的service都需要通过流程引擎来获得。
- 2、流程引擎会在创建的时候检查数据库表是否存在，如果不存在则会跑出异常，若要让引擎自动建表，则需要在配置文件添加。

3、activiti支持链式编程。



5.1、Service服务说明

5.1.1、RepositoryService仓库服务

仓库服务是存储相关的服务，一般用来部署流程文件，获取流程文件，查询流程定义信息等操作，是引擎中的一个重要的服务。

```
/** 仓库服务 */  
RepositoryService repositoryService = engine.getRepositoryService();
```

5.1.2、运行时服务

流程运行时的流程实例，流程定义，流程版本，流程节点等信息，使用运行时服务操作，是引擎中的一个重要的服务

```
/** 运行时服务 */  
RuntimeService runtimeService = engine.getRuntimeService();
```

5.1.3、任务服务

流程运行时的会产生任务，接收、办理、完成等操作使用任务服务完成，是引擎中的一个重要的服务

```
/** 任务服务 */  
TaskService taskService = engine.getTaskService();
```

5.1.4、认证服务

流程运行过程中的一些用户信息，组信息等操作使用认证服务，但是认证服务一般只作为辅助，每一个系统都有一个比较完整的人员系统

```
/** 认证服务 */  
//一般不使用自带的认证服务，每个系统都有自己的认证系统  
IdentityService identityService = engine.getIdentityService();
```

5.1.5、历史服务

流程运行时，和运行完成之后的一些历史信息，包括历史任务，历史节点等，是引擎中的一个重要的服务

```
/** 历史服务 */  
HistoryService historyService = engine.getHistoryService();
```

5.1.6、历史服务

流程运行时的任务表单信息，是引擎中的一个辅助的服务

```
/** 表单服务 */  
FormService formService = engine.getFormService();
```

总结

作业

面试题

1. 工作流的基本理论
2. Activiti简介及BPMN
3. 使用IDEA搭建Activiti开发环境
4. Activiti的整体结构
5. Process流程定义和部署维护
6. Activiti用户和用户组