Computer Science and Application 计算机科学与应用, 2019, 9(3), 650-662
Published Online March 2019 in Hans. http://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2019.93074

Hans汉斯

Design and Implementation of Affordable Housing Qualification Management System

Mengyuan Gu, Hua Ye, Yanlan Yang

School of Automation, Southeast University, Nanjing Jiangsu
Email: 1191141368@qq.com

Received: Mar. 5th, 2019; accepted: Mar. 14th, 2019; published: Mar. 21st, 2019

Abstract

In order to effectively promote the normalization and standardization of housing security qualification management and carry out real-time and efficient dynamic management in government departments, this paper designs a supportive housing qualification management system based on activiti workflow engine, which combines SpringBoot, Apache Shiro, Restful API and other technologies to reduce the difficulty of management in qualification audit and ensure a more equitable distribution of supportive housing. The system designs audit process definition with activiti workflow engine, what's more, proposes an algorithm called list-based heuristic methods for workflow scheduling to achieve task allocation, deploy process definition in Web interface. Spring MVC parses process deployment, using Apache Shiro to achieve access control. The backstage provides Restful API interface to realize reliable data interaction. The test and application results show that the security housing management system designed in this paper is stable and reliable, and achieves the expected goal.

Keywords

Affordable Housing, Qualification Management, Workflow Engine, Authority Control, Restful API

保障性住房资格管理系统设计与实现

顾梦园，叶　桦，仰燕兰

东南大学自动化学院，江苏　南京
Email: 1191141368@qq.com

收稿日期：2019年3月5日；录用日期：2019年3月14日；发布日期：2019年3月21日

摘　要

为有力地促进政府部门的住房保障资格标准化与规范化管理工作，实现实时、高效的动态管理，本文设

文章引用: 顾梦园, 叶桦, 仰燕兰. 保障性住房资格管理系统设计与实现[J]. 计算机科学与应用, 2019, 9(3): 650-662.
DOI: 10.12677/csa.2019.93074

计并构建了一种以Activiti工作流引擎为核心，结合SpringBoot、Apache Shiro、Restful API等多门技术的保障性住房资格管理系统，降低管理部门在资格审核中的工作难度，确保保障性住房更加公平的分配。首先通过Activiti工作流引擎设计审核流程定义，并提出了基于负载均衡的列表式工作流调度算法进行任务分配，然后通过Web界面部署流程定义，Spring MVC解析流程部署；采用Apache Shiro实现权限控制，后端提供Restful API接口实现可靠的数据交互。经过测试和应用结果表明，本文设计的保障性住房管理系统稳定可靠，达到了预期目标。

关键词

保障性住房，资格管理，工作流引擎，权限控制，Restful API

1. 引言

为有效解决中低收入住房困难家庭的住房问题，实现“住有所居”的目标，近年来国家不断加大保障性住房的建设。“十一五”期间全国完成了1000万套保障性住房的建设，“十二五”期间完成了3600万套保障性住房的建设计划，“十三五”期间将完成2000万套城镇棚户区住房改造。随着大规模保障性住房建设的陆续竣工，如何做好公平分配成为保障性住房政策实施的关键。公平分配是保障性安居工程的“生命线”，也是保障性住房政策落实的核心，加强资格管理，实现公平分配是对政府执政能力的一种考验。

虽然我国各省、市、地区均已建立住房保障管理信息系统，借助该管理信息系统，各省、市、地区基本实现了对保障性住房计算机化及互联网化的管理，但依然存在信息难以实现动态更新与快速共享、公示与投诉信息处理不透明等问题，对于保障性住房资格审核过程中的业务流转、明细分工、信息更新和智能决策等方面依然存在不足之处。本文针对以上问题，在SpringBoot容器框架上，提出了基于Activiti工作流引擎的资格管理方案，并在Activiti工作流引擎的基础上通过Restful API实现可靠的数据交互与信息共享，通过Apache Shiro实现严格的权限控制管理。

2. 需求分析

保障性住房分为经济适用房、中低价商品房、共有产权房、公共租赁房，申请时包括购买和租赁方式。面向的对象包括：拆迁户(集体土地、国有土地)、中低收入家庭、首次置业的无房户、新就业大学生和外来务工人员，不同类别的申请人申请保障性住房资格时所需要满足的条件和申请材料是不同的，系统需要分门别类的进行管理。

保障性住房资格管理包括资格申请管理、资格审核管理、资格复核管理、资格变更管理、资格取消管理，实现资格申请的业务流转，分别申请人提交申请后，经过街道办的初审、区房改办复审、市房改办终审后进行公示，通过公示后核发通知书，一个流程结束。每年都需要对受保对象的资格实施复核，根据情况进行资格变更和资格取消。系统需要提供流程中所需的申请表、审批表、通知书。其中申请表包括申请人基本信息、家庭成员基本信息，拆迁户需要提供拆迁的基本情况数据，中、低收入家庭需要提供资产情况，大学生需要提供毕业证书和劳动合同。审批表内容包括初审、复审和终审时各部门的意见和结果。通知书内容包括申请人姓名、申请表编号、保障性住房所在位置、建筑面积。

3. 技术综述

3.1. SpringBoot 框架

SpringBoot 使创建独立的、生产级的、基于 Spring 的应用程序变得容易。大多数 Spring 引导应用程序只需要很少的 Spring 配置。SpringBoot 框架可以创建独立的 Spring 应用程序、直接嵌入 Tomcat、Jetty 或 Underow (无需部署 WAR 文件)、提供固定的“starter”依赖项以简化构建配置、尽可能自动配置 Spring 和第三方库、提供生产就绪的特性，如度量、运行状况检查和外部化配置，不需要 XML 配置[1]。

3.2. Apache Shiro 权限控制

Apache Shiro 是一个强大且易用的 Java 安全框架，执行身份验证、授权、密码和会话管理[2]。包括三个核心组件：Subject，SecurityManager 和 Realms。Subject，即当前操作用户，Subject 代表了当前用户的安全操作。SecurityManager 是 Shiro 框架的核心，管理所有用户的安全操作，典型的 Facade 模式，Shiro 通过 SecurityManager 来管理内部组件实例，并通过它来提供安全管理的各种服务，引用多个内部嵌套安全组件形成对象图。Realm 实质上是一个安全相关的 DAO：它封装了数据源的连接细节，并在需要时将相关数据提供给 Shiro，充当了 Shiro 与应用安全数据间的“桥梁”或者“连接器”。当配置 Shiro 时，必须至少指定一个 Realm，用于认证和(或)授权[3]。Shiro 内部架构如图 1 所示。

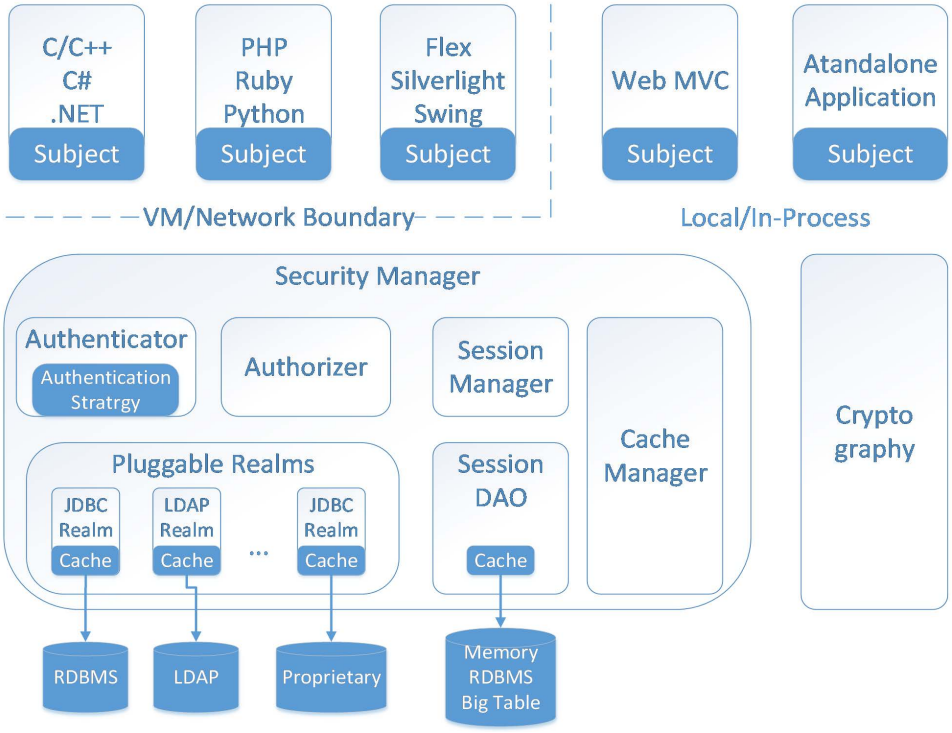

Figure 1. Internal structure of Shiro
图 1. Shiro 内部架构

应用代码通过 Subject 来进行认证和授权，而 Subject 又委托给 SecurityManager，给 Shiro 的 SecurityManager 注入 Realm 后，SecurityManager 才能对合法的用户及其权限进行判断[4]。认证和授权过程如图 2 所示。

Figure 2. Authentication and authorization process of Shiro
图 2. Shiro 认证和授权过程

3.3. Activiti 工作流引擎

对 Activiti5 是由 Alfresco 软件发布的业务流程管理(BPM)框架，它是覆盖了业务流程管理、工作流、服务协作等领域的一个开源的、灵活的、易扩展的可执行流程语言框架，是基于 Apache 许可的开源 BPM 平台。Activiti5 基于 jBPM4，与 Alfresco 的集成增加了其流程可视化与管理能力，同时通过创新的 Activiti Cycle 协作组件支持流程相关人员之间的协调，加强了集成能力[5]。

Activiti 引擎提供了七大 Service 接口，均通过 ProcessEngine 获取，并且支持链式 API 编程风格。EngineService 接口中定义了获取各种服务类实例对象的方法，提供的服务类包括[6]：

1) RepositoryService：提供一系列管理流程部署和流程定义的 API。

2) RuntimeService：在流程运行时对流程实例进行管理与控制。

3) TaskService：对流程任务进行管理，例如任务提醒、任务完成和创建任务等。

4) IdentityService：提供对流程角色数据进行管理的 API，这些角色数据包括用户组、用户及它们之间的关系。

5) ManagementService：提供对流程引擎进行管理和维护的服务。

6) HistoryService：对流程的历史数据进行操作，包括查询、删除这些历史数据。

7) FormService：表单服务。

Activiti 原生支持 Spring，可以很轻松的进行 Spring 集成，非常方便管理事务和解析表达式。Activiti 继承自 jBPM4，可以快速的读取运行时数据，仅当需要查询历史数据时再从专门的历史数据表中读取。这种设计方式可以大幅度提高运行时存取效率，尤其是当数据日积月累时依然能够快速反应[7]。

在使用 Activiti 的时候，通常情况下需要根据业务需求绘制流程文档，然后将流程文档进行部署，从而进行后续的一系列操作。部署资源无疑至关重要，如何合理使用引擎提供的部署资源功能，也是前期设计的重中之重，该过程可以检查定义的流程资源是否合理，通用以及校验流程资源格式是否正确等，从而对流程资源进行把关，规避一些不可预知的风险点[8]。流程文档部署生命周期分为四大步骤[9]：

1) 定义流程文档：客户端可以根据自己的业务需求定义流程文档。使用流程引擎的目的是为了使用流程驱动业务的流转。要用流程驱动业务就必须为业务启动一个流程实例，启动一个流程实例必须定义一系列活动，这一系列的活动就组成了一个流程定义。

2) 启动流程引擎：流程引擎启动后会自动根据构造 ProcessEngine 实例对象，这样客户端就可以通过该室离对象获取各种各样的服务类实例对象。

3) 部署流程文档：调用流程文档部署命令进行部署，该过程只需要客户端调用部署流程文档的命令即可。流程引擎收到命令之后，将流程文档中定义的元素解析为 Activiti 的内部表示 BaseElement 实例，然后对 BaseElement 实例对象再次解析，进而将其转化为流程虚拟机中的 ActivitiImpl 实例对象或者 TransitionImpl 实例对象，该过程也是将 BaseElement 实例对象注入流程虚拟机的过程。

4) 添加缓存：以上所有步骤完成之后，缓存流程定义信息，这样后续节点运转的时候，只需从缓存中取值即可，无需再次执行以上的步骤，从而大幅提升性能，流程引擎默认开启了缓存功能[10]。

3.3.1. 部署流程资源

部署流程资源的方法包括：classpath、InputStream、文本方式、zip 格式压缩包[11]。

Classpath 方式：使用 addClasspathResource 方式部署流程文档，会读取项目工程中 classpath 路径下的流程文档。使用该方式部署流程文档会使流程文档与项目产生高耦合，不建议在正式环境中使用。

InputStream 方式：使用 InputStream 方式部署流程资源需要传入一个输入流及资源的名称，直接通过类加载器获取 bpmn 文件的数据流。

文本方式：利用字符串方式可以直接传入纯文本作为资源的来源，字符串方式的实现原理是把一组字符串的内容转换为字节流后再部署。

zip/bar 格式压缩包方式：Activiti 提供了打包部署机制，可以把多个流程文档以及流程文档对应的图片或者表单等统一打包为.zip 的压缩文件或者.bar 压缩文件。

本文采用 inputStream 方式部署文件。

3.3.2. 流程文档解析

关于 XML 文档解析技术，目前经常使用的有 DOM、SAX、DOM4J 等，这些解析方式从模型上可以划分为如下两种：DOM (文档对象)模型和流模型，Activiti 选择流模型方式进行文档解析。流模型常用的技术有 SAX 和 STAX 两种。SAX 使用的是推模型，是一种事件驱动机制，文档解析的时候每当发现一个元素节点就回出发相应的事件，因此客户端需要编写监听触发事件的处理程序，使用该方式解析文档无疑增加了客户端操作的复杂度，使用起来不灵活；STAX 使用的是拉模型，文档解析时客户端可以定制自己感兴趣的节点主动从读取器中进行拉取，选择性的处理节点事件，灵活性大大提高，因此 Activiti 最终选用 STAX 方式解析流程文档。

Activiti 元素解析功能架构分为三层：

1) 元素定义层：完全交给客户端，开发人员可以结合自己的业务场景组装一系列元素完成流程文档定义工作，流程文档定义完毕之后就可以直接调用元素解析层实现流程文档的解析工作。

2) 元素解析层：负责定位流程文档、初始化元素解析器(包括子元素)、加载自定义元素解析器、查找元素解析器，最终将需要解析的元素以及属性进行解析，并封装映射为引擎中的一个个实体对象。

3) 基础支撑层：例如数据库连接管理事务管理等，Activiti 将这些公用组件抽取出来作为基础模块使用。

元素解析功能架构设计如图 3 所示。

Figure 3. Element resolution functional architecture
图 3. 元素解析功能架构

3.3.3. 工作流任务分配算法

工作流调度算法通常分为三类：1) 基于列表；2) 基于集群；3) 基于复制。

基于列表：基于列表的启发式方法根据特定的优先级机制将工作流的所有任务排序到列表中，然后按照列表中的顺序安排任务。

基于集群：基于集群的启发式方法的主要思想是通过将频繁通信的任务分为一个聚类来减少通信延迟。一般来说，基于聚类的启发式方法有两个阶段：聚类和合并。在集群阶段，原始的工作流应用程序被划分成集群，合并阶段合并集群，使集群的剩余数量等于资源的数量。

基于复制：基于复制的启发式方法通过在目标处理器上复制任务，帮助任务将数据传输到后续任务的资源。这种重复安排可以降低从任务到后续任务的通信成本。

在这三类调度中，基于列表的调度具有简单、适用性广的优点。例如，它可以有效地扩展到调度混合并行工作流或并发工作流，而其他两类启发式方法不能直接应用。基于集群的启发式算法不能处理混合的并行工作流，基于重复的启发式算法不适合并行工作流调度，因为任务重复会消耗额外的资源，从而降低了并发工作流的执行性能。因此，本文重点应用基于列表的工作流调度。

基于列表的工作流调度包括两个主要步骤：任务排序和任务分配。

在任务排序步骤中，调度算法根据特定的机制为每个任务分配一个等级值，然后根据其等级值将任务排序到一个列表中。

在任务分配步骤中，调度算法按照列表中的顺序连续地将每个任务分配到根据特定处理器选择机制执行的适当处理器。常用的任务分配方法包括：

最短工作列表(SWL)算法 SWL 算法的主要思想是将任务分配给工作列表中工作项最少的候选执行者。查找工作列表最短的执行者后返回执行者的工作列表长度。SWL 算法不需要知道执行者的任务处理时间，适用于能力表未知的情况。SWL 算法的目的是生成一个工作流日志。

最短处理时间(SPT)算法 SPT 算法主要思想是把任务分配给最优秀的执行者。它试图缩短案例处理时间，寻找任务处理时间最短的执行者。为了避免其他执行者挨饿，SPT 算法首先考虑自由执行者(没有工作项)，并且 SPT 算法对执行者的个体能力很敏感，很难平衡执行者的工作量。

最短完成时间(SCT)算法 SCT 算法将任务分配给所有候选执行者，然后计算执行者的工作列表中所有工作项的估计完成时间，找出哪个执行者首先完成任务。SCT 算法试图减少案例完成时间。

平均工作量(AVGWL)算法 AVGWL 算法的主要思想类似于 SPT 算法，同时缩短案例处理时间。与 SPT 算法不同，AVGWL 算法更注重平衡执行者的工作负载。

3.4. Restful API

REST (Representational State Transfer)是一种组织 Web 服务的架构，其目标是创建具有良好扩展性的分布式系统。如果一个系统满足了如下五条约束，那么该系统就被称为是 RESTful 的。

1) 使用客户/服务器模型。客户和服务器之间通过一个统一的接口来互相通讯。

2) 层次化的系统。在一个 REST 系统中，客户端并不会固定地与一个服务器打交道。

3) 无状态。在一个 REST 系统中，服务端并不会保存有关客户的任何状态，即客户端自身负责用户状态的维持，并在每次发送请求时都需要提供足够的信息。

4) 可缓存。REST 系统需要能够恰当地缓存请求，以尽量减少服务端和客户端之间的信息传输，以提高性能。

5) 统一的接口。一个 REST 系统需要使用一个统一的接口来完成子系统之间以及服务与用户之间的交互。这使得 REST 系统中的各个子系统可以独自完成演化。

由于系统以嵌入式集成了 Activiti 引擎，除了在内部调用引擎接口外还需要对外提供接口服务，此使 REST API 就派上了用场，只需要简单配置即可对外提供标准的 REST 服务接口，节省开发成本。

4. 设计与实现

4.1. 功能设计

系统主要分为对公服务平台和保障性住房资格管理平台。

对公服务平台主要是面向社会公众和住房保障资格申请对象。申请人可以通过门户网站时了解保障性住房的政策法规、工作动态、办事指南以及申请住房人员的公示信息等，进行保障性住房的申请并实时查看办理进度和办理结果。受保对象可以通过系统查看房源信息和分配信息，也可进行保障性住房到期退出申请。社会公众通过该服务平台了解相关信息，及时关注动态，并对公示信息进行监督和举报，从而更大程度上保证保障性住房分配的公平公正性。

保障性住房资格管理平台是系统核心功能，各种从业主体和政府职能部门通过管理平台进入，该平台需要严格的登录验证后才允许合法用户登录系统，办理住房保障业务。在验证登录时，服务器端的密码保存“加盐”后的哈希值，给每个用户分配不同的随机盐。市、区、街道住房保障部门业务人员通过资格审核管理、资格复核管理、资格变更管理、资格取消管理等子系统开展住房保障业务，实现资格业务办理、业务流转、通知书授权与打印，数据归档等功能。

4.2. 系统实现

系统的主要技术重点、难点在于保障资格审批流程如何能够自定义设定，并且简单、容易操作，做好数据规范以及数据交换方式，最关键的是如何设计任务分配算法，可以大大降低审批流程上的时间，提高工作效率。在 SpringBoot 集成环境下，采用 MySQL 作为存储数据库，引入 Activiti 工作流引擎，应用的瓶颈体现在和数据库交换数据的过程中，针对这一点选择使用 MyBatis，MyBatis 相当灵活，不会对应用程序或者数据库的现有设计强加任何影响，SQL 写在 XML 里，从程序代码中彻底分离，降低耦合度，便于统一管理和优化，并可重用。从而可以通过最优的 SQL 语句执行 Command，让引擎在速度上保持最高的性能。并通过 Apache Shiro 的权限管理严格控制相关业务人员对资源操作的不同权限，确保后一位经办人不能更改前一位的审核结果，不同部门和级别的人拥有不同的权限。技术框架如图 4 所示。

Figure 4. System technical framework
图 4. 系统技术框架

4.2.1. 数据库的选择与设计

为了提高数据的查询及数据库的性能和安全，通过一系列的处理保证信息查询速度，并且保证数据存储安全及数据库安全。采用性价比高的 UNIX 平台，配合十分成熟的 MySQL 数据库集群，为系统的灵活性和稳定性提供了基础保障[12]。根据业务需求分类，数据表分为以下几类数据：

1) 保障对象申请表：为了节约数据库的存储空间，将保障对象数据分为申请人基本信息表、同一户籍家庭成员信息表以及不同类别的保障对象申请不同类别的保障性住房所需的详细信息表三大块，避免信息冗余，通过唯一的申请人编号实现连表查询。

2) 系统信息：用户表、角色表、权限表、用户角色表、角色权限表。

3) 流程信息，相关审批流程流转信息，包括初审、复审、终审的时间、经办人和审核结果。

4.2.2. MyBatis 配置

在实际项目开发过程中，可以通过 MyBatis 定制符合保障性对象资格管理业务场景的 SQL 语句并通过自定义 Mapper 的方式执行自定义 SQL 语句。可以通过注解或配置 XML 文件方式实现数据库中申请人、申请表、审批表和通知书的查询、更新、插入、删除等操作，然后向 mybatis-configuration.xml 配置文件中注入 applicationMapper.xml 文件。如通过注解方式查询申请表中申请人相关信息：

```
String TABLE_NAME = "application";
String SELECT_FIELDS = " id, name ,applicant_type, user_id";
@Select({"select ", SELECT_FIELDS, " from ", TABLE_NAME, " where id=#{id}"})
Applicant selectById(int id);
```

4.2.3. 权限控制

创建用户、角色、权限三个实体，并配置对应的验证，以及过滤条件，通过授权模块获取用户角色和权限通过认证模块进行用户登录验证。

用户包括：普通用户、业务人员和管理员；其中业务人员的角色包括：街道办事处业务人员、区房改办业务人员、市房改办资格审核部门业务人员，需要对他们进行严格的权限控制，街道办事处业务人员只能进行资格申请受理、材料收集和录入以及申请初审；区房改办业务人员负责资格复审和通知书发

放；市房改办资格审核部门业务人员负责保障性住房资格终审，进行本市户籍人口保障性住房资格通知书打印授权，进行数据归档，房源信息发布。

4.2.4. 资格管理流程设计

绘制 BPMN 规范的流程图，如图 5 所示，并定义流程文档：

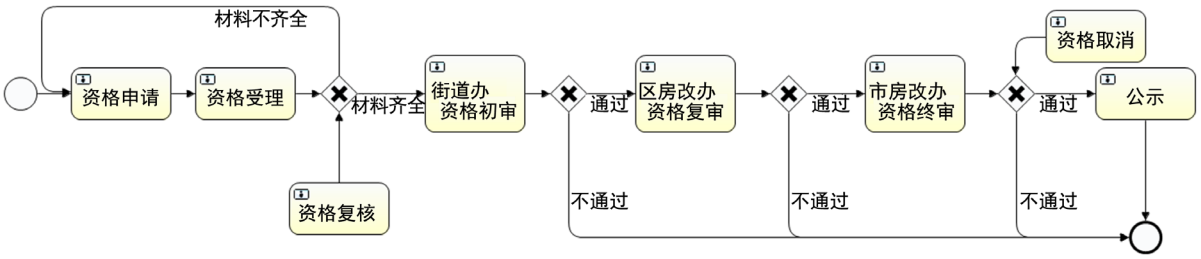

Figure 5. Qualification management process definition diagram
图 5. 资格管理流程定义图

用排他网关对流程中的决定进行建模流程执行到该网关时，按照输出流的顺序逐个计算，当条件计算结果为 true 时，继续执行当前网关的输出流。排他网关的 XML 描述如下：

<userTask2 id = "userTask2" name = "资格受理"></userTask2>

<sequenceFlow id = "flow2" name = "flow2" sourceRef = "userTask1" targetRef = "user-Task2"></sequenceFlow>

<exclusiveGateway id = "exclusiveGateway1" name = "材料是否齐全"></exclusiveGateway>

<sequenceFlow id = "flow3" name = "flow3" sourceRef = "userTask2 " targetRef = "exclusiveGate-way1" ></sequenceFlow>

<userTask3 id = "userTask3" name = "街道办资格初审"></userTask2>

<sequenceFlow id = "flow4" name = "${type1 == 1}" sourceRef = "exclusiveGateway1 "targetRef = "userTask3" >

<conditionExpression xsi:type = "tFormalExpression"><![CDATA[${type1 == 1}]]></conditionExpression>

</sequenceFlow>

<sequenceFlow id = "flow5" name = "${type1 == 0}" sourceRef = "exclusiveGateway1 "targetRef = "userTask1" >

<conditionExpression xsi:type = "tFormalExpression"><![CDATA[${type1 == 0}]]></conditionExpression>

</sequenceFlow>

采用 InputStream 方式部署流程资源，定义 approval.bpmn 的绝对路径，读取 classpath 的资源作为一个输入流，以 fileInputStream 为资源文件的输入流部署到引擎中，创建一个流程定义查询对象，然后统计已部署流程的数量并验证数量是否为 1，检查是否部署成功。

用 SpringMVC 读取已部署的流程定义列表，申明 processList()方法的访问路径，将该路径和类 DeploymentController 的@RequestMapping 注解申明的 value 拼接起来，用工具类获取 Activiti 引擎对象，创建一个视图对象并通过 listPage()方法实现分页查询，将查询结果设置到 ModelAndView 对象中，相当于调用 request.setAttribute()方法。集成到 Springboot 开发环境中，导入依赖并进行数据源和 activiti 配置

/SpringMVC 配置静态资源和直接访问的页面，添加了 thymeleaf 依赖解析视图，主要采用异步方式获取数据，通过 VUE 进行前端数据的处理和展示。启动项目，activiti 的各个服务组件就已经被加入到 spring 容器中。

4.2.5. 审核任务分配算法实现

在任务排序时，设定资格审核对象的优先级，共五级，分别为：新就业大学生优先级为一，外来务工人员优先级为二，首次置业的无房户优先级为三，中低收入家庭优先级为四，拆迁户(集体土地、国有土地)优先级为五。

在任务分配中也需要考虑执行人的负载情况。当有多位任务候选人可以进行分配任务时，应从候选人集合中选择一个当前负载量最小的候选人。假设有 n 个任务需要分配给 m 个审核员去完成，但是每个审核员手头上还有未完成的任务，且未完成的任务数不同。首先找出所有审核员中手头未完成任务数最大的审核员，然后其它审核员以该审核员的未完成任务数为标杆，计算自己可容纳的任务数，最后所有审核员可容纳的任务数之和即为总的可容纳任务数(ava_task) [13]。这里有两种情况，第一种情况是：总的可容纳任务数小于或等于 n 个待分配的任务数，此时所有审核员以最大未完成任务数为标杆，接收待分配的任务。如果刚好分配完，那么算法结束；如果还有剩下的任务未分配，那将剩下的任务抽取 m 个任务分配给每一个审核员，依次类推，直到剩下的未分配任务数小于 m 为止，然后再将这小于 m 的任务随机分配给相应数量的审核员。第二种情况是：总的可容纳任务数大于 n 个待分配的任务数，此时降低一个单位的标杆，然后循环计算可容纳的任务数，直到退出循环(循环终止条件为：可容纳任务数 − 分配任务数 ≤ 低于当前标杆的审核员数)。

此时，算法达到了任务负载均衡的效果，但没有考虑到审核人员处理任务的能力和所需要的时间。因此，在任务分配过程中，本文采用基于负载均衡的列表工作流启发式调度算法。把任务分配给最优秀的执行者，缩短案例处理时间，寻找任务处理时间最短的执行者。为了避免其他执行者挨饿，首先考虑自由执行者(没有工作项)，同时注重平衡执行者的工作负载。计算所有执行者的平均工作量，返回所有执行者的工作列表，然后接收工作负载低于平均值且任务处理时间最短的执行者[14]。

算法输入：PL，候选执行者列表，每个执行者都有自己的工作列表(WL)；AT，所有候选执行者的能力表。

```
1: best_ability ← MAX_DOUBLE
2: best_performer ← NONE
3: avg_complete_time ← avergerTime(PL,AT)
4: for all performer P ∈ PL do
5: if est(P.WL,AT) < avg_complete_time then
6:  if best ability > getAbility(P,AT) then
7:        best ability ← getAbility(P,AT)
8:        best performer ← P
9:        end if
10:       end if
11: end for
12: return best performer
```

为了实现工作量负载均衡且尽可能缩短审核处理时间，作如下改进：若总的可容纳任务数小于或等于 n 个待分配的任务数，此时所有审核员以最大未完成任务数(max_task)为标杆，根据返回的工作负载低

于平均值且任务处理时间最短的执行者列表进行任务分配。如果刚好分配完，那么算法结束；如果还有剩下的任务未分配，重复上述分配步骤直到所有审核员达到最大未完成任务数。

假设有 8 个审核员，未完成任务和审核能力表如表 1 所示。

Table 1. Unfulfilled tasks and audit capability statement
表 1. 未完成任务和审核能力表

	审核员 1	审核员 2	审核员 3	审核员 4	审核员 5	审核员 6	审核员 7	审核员 8
未完成任务	1	4	3	5	2	7	2	1
审核时间(min)	100	60	80	80	60	90	90	120

负载均衡条件下，随着任务数增加算法优化前后完成时间对比如表 2 所示。

Table 2. Comparison to completion time before and after the improvement of the algorithms
表 2. 算法改进前后完成时间对比

新增任务(个)	完成时间(min)	
	优化前	优化后
20	1750	1630
40	3570	3320
80	6930	6500
160	13,730	11,040
500	42,640	38,500

4.2.6. 集成 RESTAPI

利用现有模块(activiti-rest.war)代替直接 API 调用，各个系统根据 rest 模块的接口规范访问 REST 资源，实现统一处理。

将 activiti-rest.war 解压到 Web 服务器的应用部署目录，修改 activiti-rest/WEB-INF/classes/db.properties 里面的数据库配置后，将引擎配置文件 Activiti-context.xml 和引擎数据库属性配置文件 db.properties 复制到资源根目录中，接下来要在 web.xml 中加入 REST API 的相关配置，包括 Activiti 引擎上下文初始化监听器(ActivitiServletContextLisener)以及 Servlet 映射。之后启动应用，通过 Spring MVC 提供的 RestTemplate 测试是否部署成功可以正常的提供服务。

4.2.7. 绘制 Web 界面

前端采用 Vue.js 渐进式框架构建用户界面，采用自底向上增量开发的设计，实现复杂单页应用要包括 home、login、apply 和 approval 四个模块，在 routers.js 中配置 VUE 文件的路径[15]。

资格审核界面如图 6 所示。

Figure 6. Interface of qualification examination
图 6. 资格审核界面

5. 结束语

本文以规范管理、优化流程为目标，开发了保障性住房资格管理系统，该系统采用 Java 语言在 Intellij IDEA 2017 集成开发环境中实现，嵌入式集成 Activiti 引擎，实现保障性住房资格管理的工作流程控制，着重研究了 Activiti 工作流引擎任务分配算法，考虑了业务人员能力因素，提出了基于负载均衡的工作流引擎启发式算法，达到了均衡任务数量并缩短审核时间的目标。经测试系统功能和性能指标都满足设计需求，从系统实际运行效果来看，各项指标正常，达到了预期目标，提高了保障性住房资格管理的效率。后续还会在此基础上继续深入研究和开发，例如提供智能决策服务，实现保障性住房资格预判等功能，从而更大程度上优化流程，提高公平分配效率。

参考文献

[1] Suryotrisongko, H., Jayanto, D.P. and Tjahyanto, A. (2017) Design and Development of Backend Application for Public Complaint Systems Using Microservice Spring Boot. *Procedia Computer Science*, **124**, 736-743. https://doi.org/10.1016/j.procs.2017.12.212

[2] 宋成明. 基于 Shiro 的某高校科研信息管理系统的设计与实现[J]. 智能计算机与应用, 2017, 7(4): 62-63, 66.

[3] 许滔. 基于 Shiro 的移动应用权限控制系统的设计与实现[J]. 现代计算机(专业版), 2016(6): 97-100.

[4] 翁云翔. Java 安全框架 Shiro 在 Web 中的研究与应用[D]: [硕士学位论文]. 武汉: 武汉邮电科学研究院, 2016.

[5] 吕俊瑞, 陈波, 叶承卓. Activiti 工作流在物流业务系统中的应用[J]. 攀枝花学院学报, 2018, 35(5): 83-88.

[6] 夏亮. 基于 Activiti 的铁路生产管理系统的设计与实现[D]: [硕士学位论文]. 北京: 北京交通大学, 2018.

[7] 闫洪磊. Activiti 实战[M]. 北京: 机械工业出版社, 2014: 5-6.

[8] Wei, H., Ge, J., Li, C., Li, Z., Lei, M. and Hu, H. (2017) Flexible Manufacturing Chain: A SCM for Electronic Commerce Enterprise in Clothing Industry Based on Activiti. *Communications in Computer and Information Science*, **23**, 3-14. https://doi.org/10.1007/978-981-10-3996-6_1

[9] Baina, K. and Baina, S. (2013) User experience-based evaluation of open source workflow systems: The cases of Bonita, Activiti, jBPM, and Intalio.

[10] Yang, S.L. and Hu, J.P. (2015) Design of Task Workflow Based on Activiti Technology. *Applied Mechanics and Materials*, **3822**, 54-60.

[11] 谷占忠. 基于 Activiti 的行政事业单位预算精细化管理系统的设计与实现[J]. 电脑知识与技术, 2018, 14(3):

101-102.

[12] ScaleGrid. (2018) MySQL Hosting on Azure, Fully Managed Cloud Database Service Launches at ScaleGrid.

[13] 喻神. 一种基于平均思想的任务分配算法[EB/OL]. https://blog.csdn.net/y1250056491/article/details/77652739

[14] Xu, J., Huang, Z., Yu, Y. and Pan, M. (2012) A Performance Analysis on Task Allocation Using Social Context. *Second International Conference on Cloud and Green Computing*, Xiangtan, 14 February 2013, 637-644.

[15] 旷志光, 纪婷婷, 吴小丽. 基于 Vue.js 的后台单页应用管理系统的研究与实现[J]. 现代计算机(专业版), 2017(20): 51-55.