上海市村庄规划编审平台的实现

摘要

为了加快上海市城乡一体化建设和发展，切实地对本市村庄规划实行信息化管理，需要构建一个完整的线上审批流程。同时根据人口和已有规划分布，平衡土地利用，部署各类公共设施，引导布局，建设契合当地村庄特点的，与当地经济水平发展相辅相成的村庄环境。

村庄规划的审批流程包括计划上报，案件审批和成果入库几个主流程，主流程下分别又包含多个子流程（节点），需不同角色的用户进行协同审阅从而决定该规划是否可行，或做出对应的计划调整。同时用户也能在系统中查看相应计划的审批流转情况，从而跟进纸质文件的传递。

本论文针对万达信息政府服务事业部国土团队所负责的上海市规土局项目需求，为解决相关处室无信息化系统支持，行政审批效率低下等问题，决定根据现有应用开发技术（nutz，jquery，ajax等）和参考以往的行政审批系统开发经验，编写出一套符合村庄规划审批业务特点的，基于工作流模型和权限管理的编审平台。

本平台旨在设计一个便于维护和扩充审批流程和权限管理的框架，提高案件编审的统一性、协同性、准确性，同时能为工作人员在一定程度上带来便利，使工作效率得到提高，加强人机交互，满足日益增长的规划信息化管理需求，也可以为加快城市化建设进程提供一定的帮助。

关键词：nutz，审批系统，规划信息管理，工作流

**IMPLEMENTATION OF THE PLANNING**

**AND AUDIT PLATFORM FOR VILLAGE PLANNING IN SHANGHAI.**

**Abstract**

In order to accelerate the construction and development of urban-rural integration in Shanghai, it is necessary to construct a complete online audit process for the implementation of informatization management of the village planning in this municipality. According to the population distribution and existing planning, balance the land use, deploy all kinds of public facilities, guide the layout and construct the village environment with the characteristics of the local village and helpful for local economic development level supplement.

Village planning audit main process including ‘planning report’, ’case audit’, ’result entry’. There are multiple sub-processes (nodes) under the main process, the users of different roles will audit together to determine whether the plan is feasible or to make a corresponding plan adjustment.

This thesis aims at the project requirements of Shanghai Planning and land resources administration, which is responsible for the land team of wonders group.

To solve the problem such as related departments didn’t have information system support,and efficiency of administrative audit is very low. We decided according to the existing application development technology (nutz, jquery, ajax, etc.) and the reference of administrative examination and approval system development experience,code a platform which accord with the characteristics of village planning business,based on workflow model and authority management.

This platform is to design an audit framework which is easy to maintain and expand audit process and authority management. Improve the unification, synergy and accuracy of the case compiling. This platform also can bring convenience for staff to a certain extent, make working efficiency gets improved, strengthen man-machine interaction, meet the increasing demand planning information management, also can offer some help to accelerate the urbanization process.

**Keywords**: nutz，audit system，planning information management，workflow

**1 绪论**

* 1. 论文背景

如今社会的发展已经走向了高速的信息化时代，包括政府部门的市政建设类工作，例如医疗健康、保险养老、文化教育等各领域都离不开现代信息科技的管理手段。国土资源的管理也是政府在当前时代背景下所关注的重点。

村镇建设一直是各地区经济发展的不可或缺的一部分，目前各地区的村镇建设无信息化系统支持，其编制审批工作主要依靠纸质文件进行流转，一方面行政审批效率低下，另一方面不利于信息的管理和利用。

为了优化我市村庄规划现有业务流程，实现网络化、无纸化办公，自动沉淀项目管理的各项流程数据和规划成果数据，规范工作流程、提高工作效率、加大信息利用程度和监管力度，村庄规划编审平台应运而生。

* 1. 论文的主要任务

本论文的主要任务围绕通过现有的web开发技术搭建一个线上的村庄规划审批平台，该平台主要满足一下几个要求：

1. 采用年度计划管理模式，保证各区及全市各项村庄规划编制的计划汇总和统一指导。
2. 村庄编制审批过程中涉及到的相关工作流程串联起来，形成完整的审批系统程序，实现网络化和无纸化审批。
3. 案件在各个环节，用户都可以对其流转情况进行跟踪，查询该案件的办理进度。
4. 能够在平台上直接生成相关审批文件的电子版，减少工作人员填写纸质文件的工作量，也方便在平台上进行文件管理。
   1. 论文难点
5. 建立一个灵活可扩展的工作流模型：

审批系统中最重要的一项流程就是各级审批，如果要拿一个数据结构来模拟的话，有向图(拓扑图)是非常合适的参照模型了。

通过有向图不仅能提供一个流程推进的轨道，还能够满足不同情况下走不同路径的选择性和灵活性。尽管审批流程一般是线性流转的，但是也会有跳过某些（可选）节点直接到达后面节点的需求，这时候根据有向图的特性，代码能够帮助用户分析出可到达的下一节点会有哪些，这些节点一般会按系统术语区分（一般为：默认、流转、特送和回退），用户可以自主选择按那种类型推送工作流，关于以上节点类型将在后面的章节进行说明。



1. 建立一个完善的权限管理体系

权限管理是每个系统都会有所考虑的一个关键部分，审批系统更是要对权限进行相应的细分。因为每一个流程节点的经办人是跟一个业务角色进行绑定的，该业务角色不仅有编写该节点的权限，还会被分配一些其他功能权限，比如说增删改查某项业务表的权限。

关于权限管理，设计的方式会有很多，但是要从直观和易于管理两点出发还需要一个清晰的设计思路。在此参考了一些系统在权限管理部分的设计经验，oracledb的权限设计是一项重要的借鉴来源，它将用户、角色、权限按照三个模块进行区分，在降耦的同时建立相互的多对多关系，使得两两之间的多对多关系变得非常直观，也更易于管理。

1. 直观的数据化展示

每一条在平台中创建的计划都能在系统中查询到，如何让用户能够获取到用户最想获取的数据是是一项关键问题。无论是采用图表形式还是采用筛选字段的方式展示列表，都是不错的展示数据的方式。只要能满足用户的查询需求（提供相应的查询条件和筛选结果）以及展示准确的数据，就算达到我们数据展示直观化的目的。

* 1. 论文创新之处

1. 采用nutz开发框架，一款国人开发的web开发框架，以最大限度的提高Web开发人员的生产力为目标，提供各种常见的web开发功能：Dao、Ioc、Mvc等。
2. 自定义的工作流模型。
3. 对上海市的村庄规划信息进行管理，其管理功能包括案件办理，对流转中的案件进行审批并填写办理意见；新增计划，区县经办人将该区的村庄规划进行汇总并新建，然后提交给市局的经办人进行审理审核；案件查询，根据用户的查询条件加载查询结果，并能根据筛选结果查询案件的详细信息；系统管理，将各个模块的管理权限提供给不同角色用户，让用户自主管理负责的模块。
4. 提供字典管理，用可以根据字典查询相关角色字典，流程字典，权限字典的定义从而理解业务术语。
5. 简单的审批文件管理。
6. 日志记录（nutz内置）。
7. 简单的页面重构及js初始化。
   1. 论文结构
8. 绪论，阐述了本论文的背景，主要任务就相关难点以及克服这些难点的同时展现的创新之处/
9. 介绍需要的技术、所使用的编程语言、项目所使用的开发框架和开发工具。以及部署环境时可能产生的问题和遇到的困难。
10. 对系统的需求进行分析（包括功能需求、界面需求和一些非功能需求），首先介绍总体功能，并列出相应的功能一览表，然后详细介绍对应的功能，之后再简略地介绍界面需求和非功能需求。
11. 从系统设计的角度进行介绍，介绍项目的功能模块，同时说明启动设置和配置文件作用。
12. 从数据库设计的角度结合前一章系统设计的说明，介绍系统架构原理及分层方案，以及解释查询sql或框架自带的sql包装类使用方式。
13. 主要讲了系统的贡献和未来进一步提高的方向。
14. **开发技术及方法**
    1. 系统开发语言
       1. JAVA

Java是web应用开发中应用最广发的语言之一，从早期的servlet，到后面的jsp，再到后来的各类开发框架，java语言早已形成自己一套完整的开发体系，无论是大型的企业级系统开发，还是小型的个人站点或门户博客之类的开发，它都能一门方面快捷的语言驱动的业务逻辑的正常运转。

如今java的版本已经迭代更新到10，在巩固原来面向对象编程的方式下，不断添加更多实用的功能。在基于组件，具有平台无关性的J2EE 结构下使得J2EE 程序的编写十分简单，开发人员不用考虑服用问题，只需集中精力于业务逻辑的实现。

* + 1. SQL

结构化查询语言SQL（STRUCTURED QUERY LANGUAGE）是最重要的[关系数据库](https://baike.baidu.com/item/%E5%85%B3%E7%B3%BB%E6%95%B0%E6%8D%AE%E5%BA%93)操作语言，并且它的影响已经超出[数据库](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%BA%93)领域，得到其他领域的重视和采用，如[人工智能](https://baike.baidu.com/item/%E4%BA%BA%E5%B7%A5%E6%99%BA%E8%83%BD)领域的数据检索，第四代软件开发工具中嵌入SQL的语言等。

通过将SQL语言和相关的DAO层封装工具结合，编写相关的DQL和DML语句操作数据库已经成为了最主要的数据库查询和修改方式。利用这种修改方式不仅能够使代码结构变得更加清晰，还能提高程序的复用性，提高代码编写效率。

* 1. 系统开发框架

本系统使用的开发框架主要有两种：

1. Nutz框架，主要负责后台的开发。Nutz框架是一个由国人开发，国人维护web开发框架。

不同以往常用的ssh和ssm框架，nutz框架只需导入一个jar包，即可实现spring的ioc、aop功能；hibernate和mybatis的数据库实体映射功能；以及springmvc或struts的mvc功能。

由于只需导入一个jar包，用户即可免去不同框架之间的兼容性测试，也不用再重复检查每种框架是否与当前版本的java语言相兼容。

1. Bootstrap框架，主要负责前台页面的展示和美观。Bootsrtap也是广受程序员喜爱的前段框架，它是基于 HTML、CSS、JAVASCRIPT 的。再提供美化页面标签元素的同时，还能提供各种组件类似于创建图像、下拉菜单、导航、警告框、弹出框等等。开发人员只需简单的声明或绑定操作即可调用。
   1. 本章小结

本章围绕了系统开发使用的技术做了简要描述，使用的语言主要是java其中可能还会包括一些html或者css之类的前台代码和sql之类的脚本语言。使用的框架按前后区分主要是bootstrap和nutz框架，每种框架各司其职完整地展现了该平台所具备的功能。

1. **系统的需求分析**
   1. 功能需求
      1. 功能一览表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **一级需求** | **二级需求** | **操作频率** | **优先级** | **备注说明** |
| 新建计划 | 计划新增 | 中 | 高 |  |
| 计划修改 | 中 | 高 |  |
| 计划删除 | 低 | 高 |  |
| 计划启动 | 中 | 高 |  |
| 计划查询 | 高 | 高 |  |
| 业务办理 | 基础准备 | 高 | 高 |  |
| 方案审批 | 高 | 高 |  |
| 成果入库 | 高 | 高 |  |
| 案件查询 | 信息监控 | 高 | 高 |  |
| 签收回收 | 中 | 中 |  |
| 综合管理 | 用户管理 | 中 | 高 |  |
| 人员管理 | 中 | 高 |  |
| 流程管理 | 高 | 高 |  |
| 字典管理 | 低 | 中 |  |

注：

1. 操作频率：分为高、中、低三级（代表功能的操作频度），项目可根据客户的需要进行调整
   1. 如每日都需要大量执行的业务功能，级别为“高”；
   2. 每日/每周定期进行的汇总统计功能，级别为“中”；
   3. 事件触发型的功能，级别为低
2. 需求优先级：分为高、中、低三级（代表需求的可实现顺序）。
   1. 高：说明是一个关键任务的需求，必须在这一个版本中实现；
   2. 中：说明这个需求是系统所最终需要的，但如果有必要的话，可以延迟到下一个版本中实现；

低：说明这个需求是对业务流程/功能/质量上的增强，如果资源允许的话，实现这些需求会使产品更加完美。

* + 1. 新建计划
       1. 计划新增

区县村镇科经办人将规划基本信息（包括规划名称、区县、规划环节、村庄名称、规划面积）、设计单位信息（单位名称、联系人、联系方式）以及规划的预计办结时间填写并新增。

* + - 1. 计划修改

区县村镇科经办人新增村庄规划计划之后，可以在计划列表中找到自己新增的计划，并在操作栏找到修改按钮进行修改。可以修改的字段除了区县和规划环节，其他均可修改。具有修改权限的角色除了区县经办人，市局审理案件的村镇处经办人同样具有修改上报案件的权限。

* + - 1. 计划删除

区县村镇可经办人新增村庄计划后，如果有删除案件的需求，可以在对应列表的操作栏中找到删除按钮，并将案件删除，此删除权限的只开放给区县经办人，村镇处经办人不具有删除案件的权限。

* + - 1. 计划启动

村镇处经办人在协同领导审理案件后，认为该规划具备启动审批流程的条件，可以在对应列表的操作栏中点击启动按钮，即正式开始案件的审批流程。此启动权限只开放给村镇处经办人。

* + - 1. 计划查询

用户可在新增计划菜单下，显示计划列表，如果用户的区县划分为市局用户，则显示全市新增的村庄规划计划，如果用户划分为非市局用户，则显示对应区县申报的村庄规划计划。

* + 1. 业务办理
       1. 基础准备

基础准备环节为流程默认的起始环节。用户可以在该审批环节下做一些基础准备的信息填写，文件上传和申请等操作。

* + - 1. 方案审批

方案审批环节为流程的第二环节，用户可以在该审批环节下填写审批时的办理意见。

* + - 1. 成果入库

成果入库主要分为归档和入库两个操作步骤，审批通过的案件将进行归档，形成的成果文件将入库以便于日后进行汇总和管理。

* + 1. 案件查询
       1. 信息监控

用户可以在信息监控查询到所有的村庄规划信息，并能够按一些主要的筛选条件（区县、年份、是否办结）进行筛选。同时用户还能点击操作栏中的查看按钮查看案件的详细信息（办理意见、流转信息等）。

* + - 1. 签收回收
         1. 签收

当案件流转到相关的用户下时，用户必须要签收才能进行审批操作，签收界面下显示所有与当前用户相关的未签收的案件。

* + - * 1. 回收

当用户提交案件之后，下一节点的经办人未签收案件时，用户可以在回收列表中找到自己提交的案件，并进行回收重新审批办理。

* + 1. 综合管理
       1. 用户管理

用户管理可以查询已经在系统中注册的用户。

* + - 1. 人员管理

人员管理可以查询在系统中已经添加的人员，并提供修改新增和修改人员信息的功能。

* + - 1. 流程管理

该管理界面下，用户可以对流转中的案件的流程信息进行纠正和调整。同时可以查看已完成的历史流程记录。

* + - 1. 字典管理

用户可以查看所有字典表的定义及字典表下对应的记录。

* 1. 界面需求
* 系统界面使用bootstrap前段框架进行美化；
* 系统各模块界面框架内容风格保持一致；
* 输出设备：显示器、打印机等；
* 显示风格：主流浏览器界面风格；
* 显示方式：分辨率自适应。
  1. 非功能需求
     1. 性能需求
        1. 响应时间
* 查询与统计响应时间<3秒。
* 表单录入功能响应时间<5秒。
* 提交流程响应时间<2秒。
  + - 1. 吞吐量
* 支持系统最大在线人数至少200人。
* 支持系统最大并发人数至少20人。
  + 1. 可用性
* 风格简约，色调柔和，合理利用浏览器空间，减少不必要的操作。
* 易用性强，从操作者的角度出发，提升用户体验效果。
* 兼顾效率，合理减少用户操作的等待时间。
* 友好提示，在人机交互过程中，有效甄别输入的正确性。

1. **上海市村庄规划编审平台系统设计**
   1. 概要设计

“上海市村庄规划编审平台”包括计划管理、规划审批、工作流管理、用户认证管理、查询统计五个模块，本论文负责全部五个子模块的设计和实现。

1. 计划管理：计划的增、删、改、查及启动该计划的审批流程。
2. 规划审批：推进计划的审批流程并填写相关的审批意见。
3. 工作流管理：工作流作为案件推进的凭证及推进情况的追踪来源。在已有的流程字典下，用户可以根据流程拓扑图选择流程的前进方向。
4. 用户认证：用户认证涉及角色、权限、用户和账户四个对象之间的相互绑定关系，具有用户管理权限的操作人可以对这些对象进行绑定和解绑。
5. 查询统计：对案件进行条件筛选，并能根据筛选结果获得相应的案件的详细信息。
   1. 功能设计

功能模块图

* + 1. 计划管理模块

计划列表，如图4-1



**图4-1 计划列表界面**

区县经办人点击页面右上方的修改“新增计划”按钮可以进入到新增计划界面，如图4-2。



**图4-2 新增计划界面**

用户点击规划环节栏的选择按钮，可以加载出环节列表如图4-3，用户可以根据实际的情况选择需要进行的环节，而未选的环节在审批过程中可以跳过。



**图4-3 环节列表**

同时在选择村镇的时候，也是通过点击村镇栏的选择按钮加载对应区县（跟当前用户的区县相关）的区县列表，如图4-4。



**图4-4 村镇列表**

当用户填写好所有相关信息之后，点击右下角的新建计划按钮，即可完成新增计划步骤。计划列表中会显示新增的计划，用户也可以根据年度进行计划的筛选并点击修改按钮修改相关的计划信息，修改界面如图4-5所示。



**图4-5 计划修改界面**

用户修改后可以点击右下价的保存按钮保存修改后的计划信息。

若想撤销计划的上报，新建计划的经办人或与创建人同属同一区县的经办人可以进入在计划列表界面，点击操作栏中的删除按钮，删除对应要撤销的计划。如图4-6。



**图4-6 删除计划**

市局村镇处对计划对修改后确认的信息进行审理，如果满足纳入规划计划的条件，可以点击列表右侧操作栏中的启动计划进行启动，此时计划的状态会由未流转变成流转中。如图4-7。



**图4-7 启动计划**

* + 1. 规划审批模块

审批案件列表，列表的列为流程环节，行为案件记录，如图4-8所示。



**图4-8 审批案件列表**

办理人员点击列表中对应的节点会进入相应的审批界面，审批界面如图4-9所示。



**图4-9 审批界面**

若案件尚未签收，用户需先点击签收按钮完成签收操作，根据案件基本信息，填写办理意见，完成对应节点的审批流程。

同时用户也可在导航栏中菜单中的案件查询→待签收案件中，查看自己待签收的案件，并选择对应的案件进行签收，如图4-10所示。



**图4-10 待签收案件列表**

用户签收案件后，在案件审批界面，用户填写办理意见后点击保存按钮，可保存当前节点的办理意见，如图4-11所示。



**图4-11 保存办理意见**

当用户填写好办理意见之后，可以点击下方的提交按钮，进入提交界面，如图4-12所示。



**图4-12 提交界面**

关于提交界面的节点和办理人员的分类和选择，将会在详细设计中进行进一步说明。

选择好节点和人员之后，点击页面下方的确认按钮，即可流转到下一节点的，并由下一节点对应办理人员进行审批。

审批页面下方的流程信息按钮，用户点击后可跳转到案件的详细信息界面，本文将会在查询统计模块进行界面的说明。

* + 1. 查询统计模块

案件查询列表，如图4-13所示。



**图4-13 案件查询列表**

用户可以根据系统所提供的筛选条件对案件进行自定义查询，如图4-14所示。



**图4-14 案件查询列表**

用户可以根据筛选结果点击对应案件操作栏中的查看按钮查看案件信息。案件详细信息如果图4-15所示。



**图4-15 详细信息界面**

详细信息界面又分为四块子标签页：

1. 案件信息：显示新增计划时填写的基本信息，如图4-15。
2. 办理意见：显示案件审批流程中，各节点办理人员填写的案件信息。如图4-16所示。



**图4-16 办理意见界面**

1. 流程信息：上方显示当前流程信息，下方显示历史流程信息列表。如图4-17所示。



**图4-17 流程信息界面**

* + 1. 用户认证模块

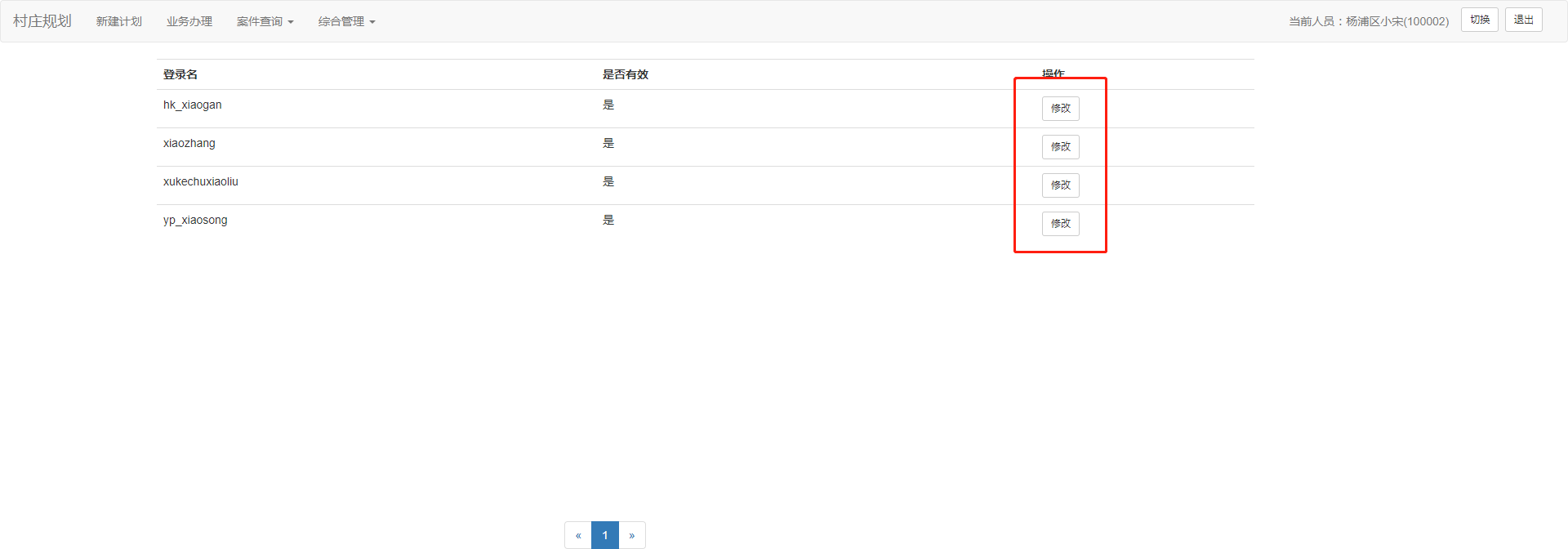
用户认证在系统上主要涉及两块菜单，一块是用户管理部分，一块是人员管理部分，如图4-18所示。



**图4-18 用户认证菜单**

1. **用户管理**

用户管理界面，管理员可以查看或修改相关用户的信息，如图4-19所示。



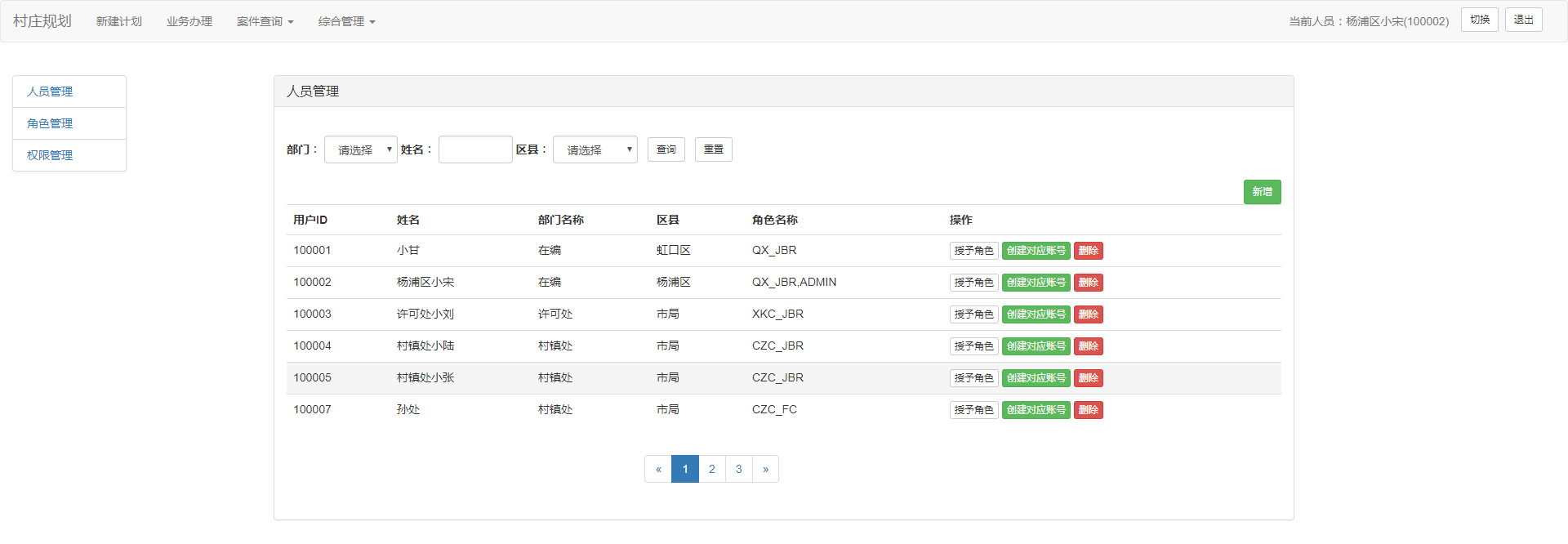
**图4-19 用户管理界面**

1. **人员管理**

人员管理分为以下三个部分，每个部分之间负责不同的与权限相关的实体管理，管理员可以负责他们绑定和解除他们之间的关联关系。

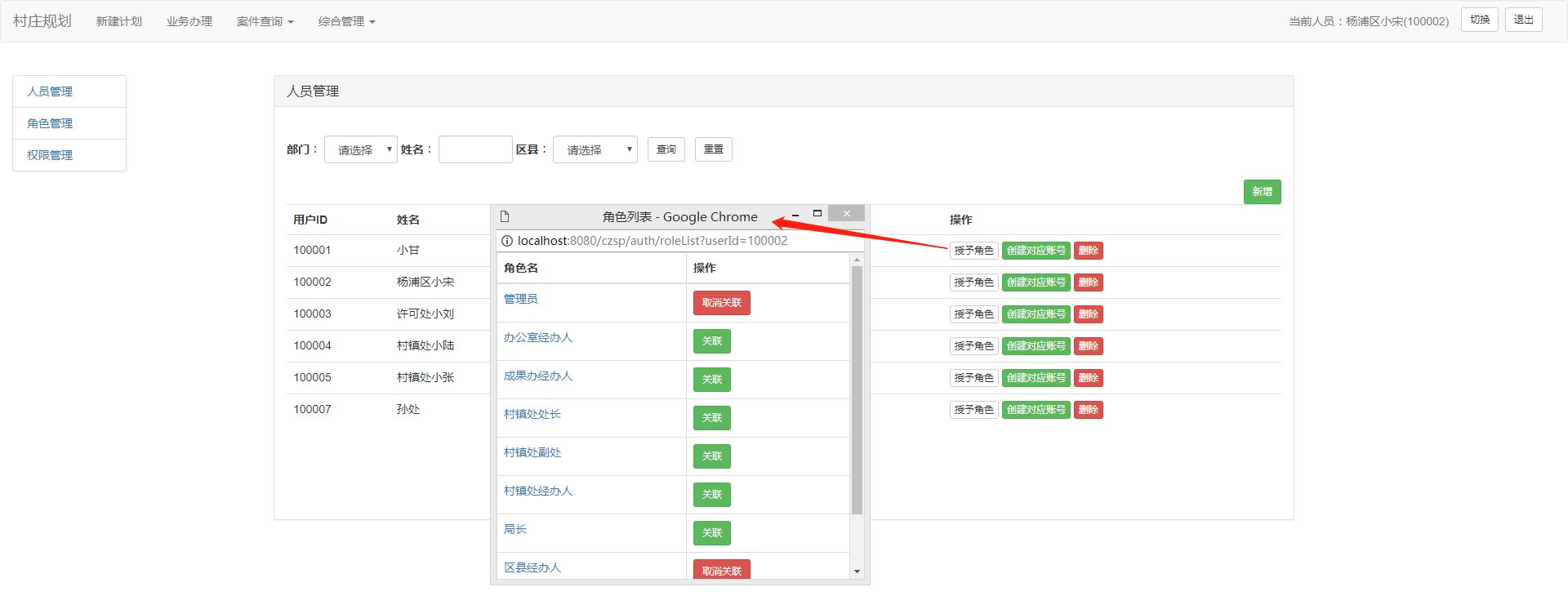
* 1. **人员管理**

人员管理界面，如图4-20所示，用户可以根据相关的检索条件查询到对应的用户，并在操作栏可以进行“授予角色”，“创建对应账号”，“删除”的用户操作。



**图4-20 人员管理界面**

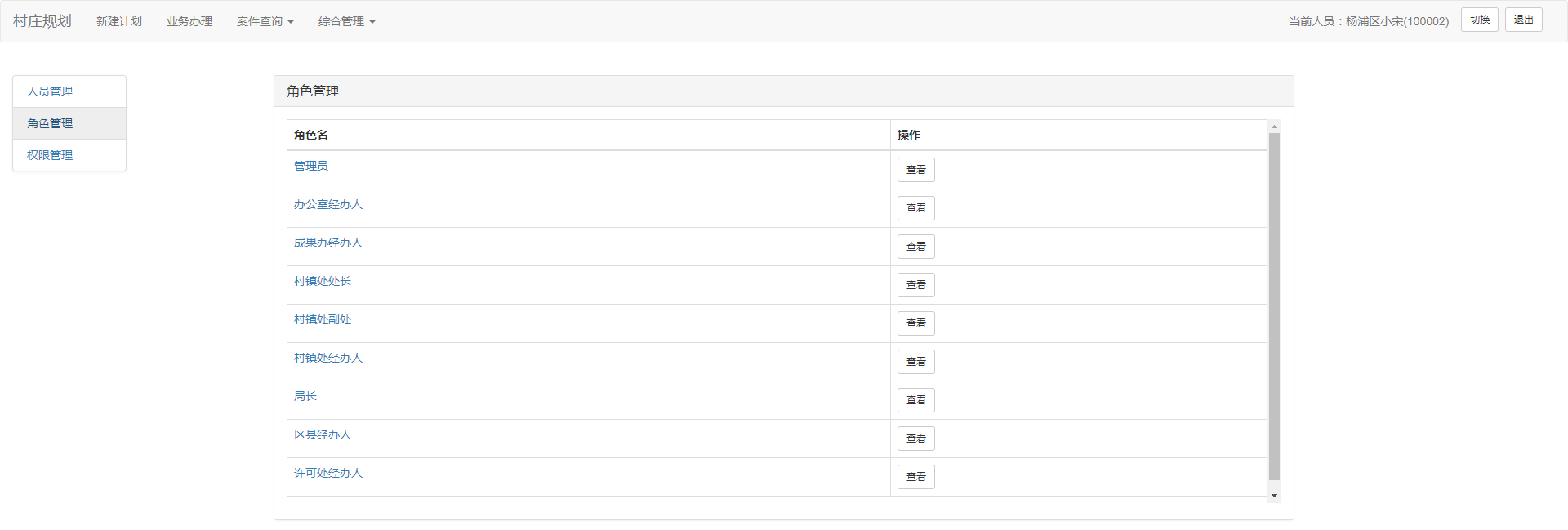
授予角色：点击授予角色操作按钮，会进入相应账号的角色绑定界面，如图4-21所示，如果某种角色已经与选择的人员关联，用户可以点击“取消关联”按钮，取消对应角色的关联；如果未绑定，可以点击关联按钮进行人员和角色的绑定。



**图4-21 角色绑定界面**

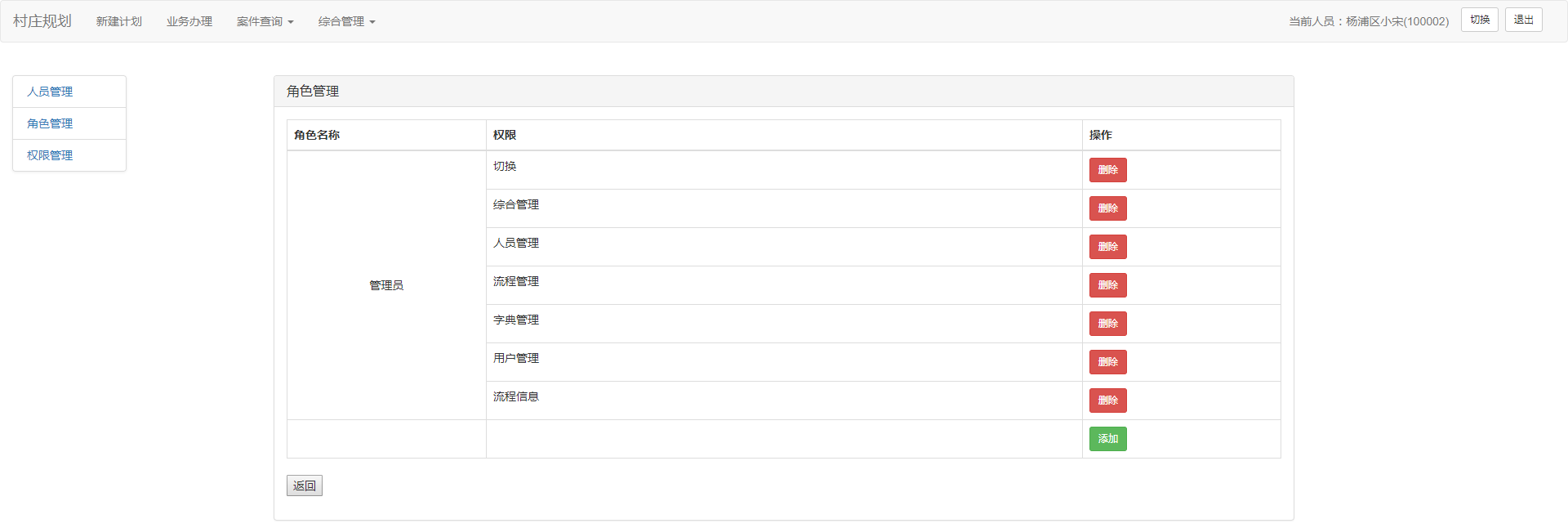
* 1. **角色管理**

角色管理界面，如图4-22所示，用户可以在该界面在权限和角色之间进行相互绑定。

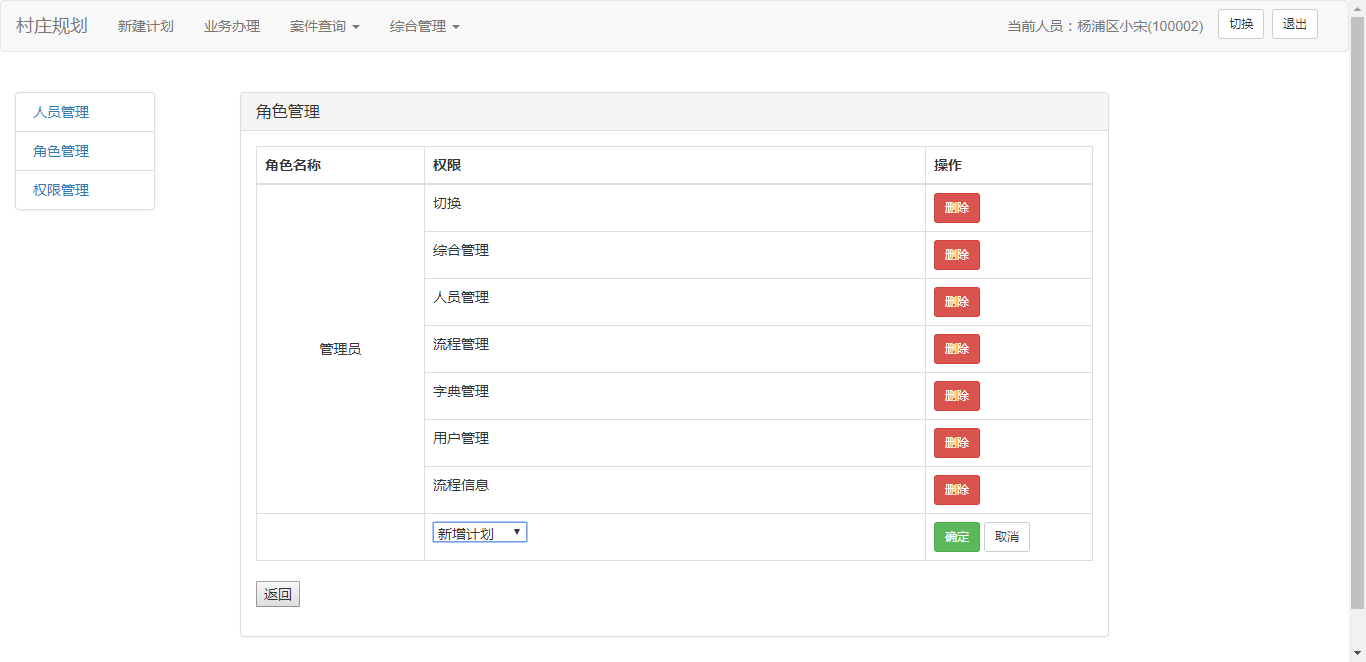


**图4-22 角色管理界面**

当用户点击相关的角色时，会转到角色-权限之间的关联界面，如图4-23所示。在此界面，用户可以给选中的角色添加权限对象，如图4-24所示，同时也可以删除已经关联的权限对象。

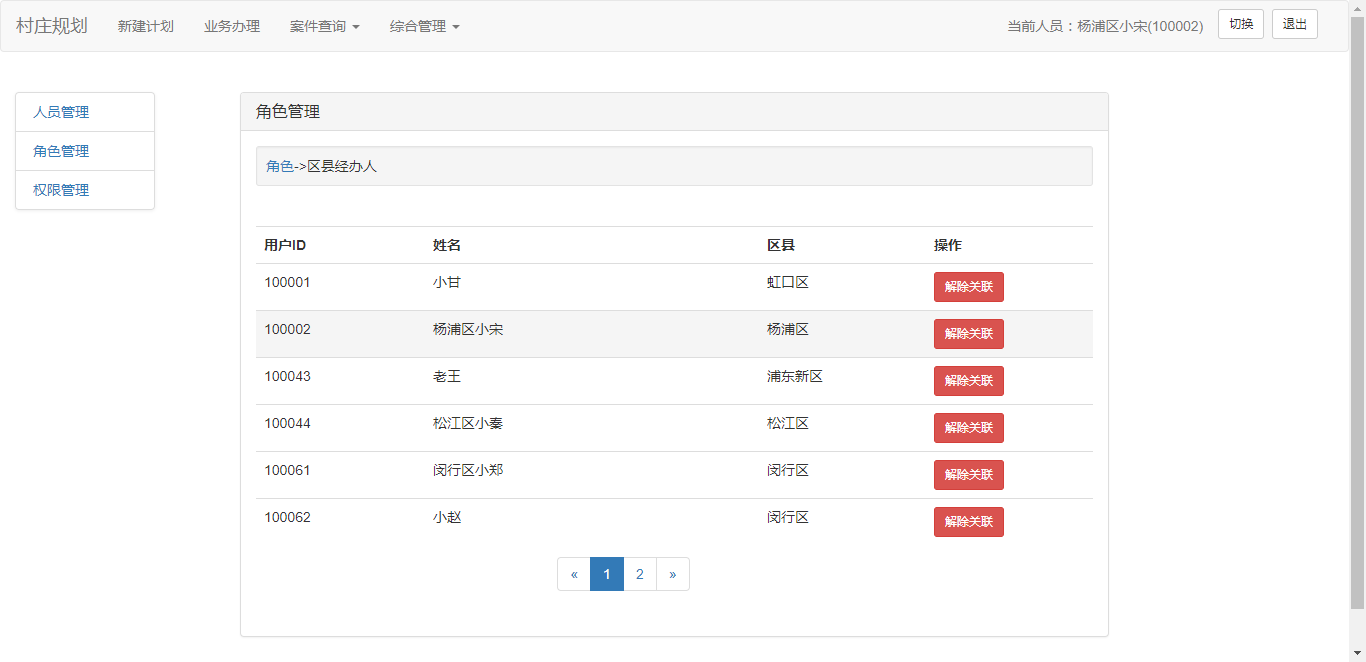


**图4-23 角色-权限管理界面**



**图4-24 添加权限**

当用户点击某一角色操作栏的查看按钮，可以查看所有具有这一角色的的用户，如图4-25所示。用户可以点击右侧的取消关联按钮，取消该用户与选中角色的关联。如要重新绑定，需要在人员管理中选择角色进行绑定。



**图4-25 角色-用户界面**

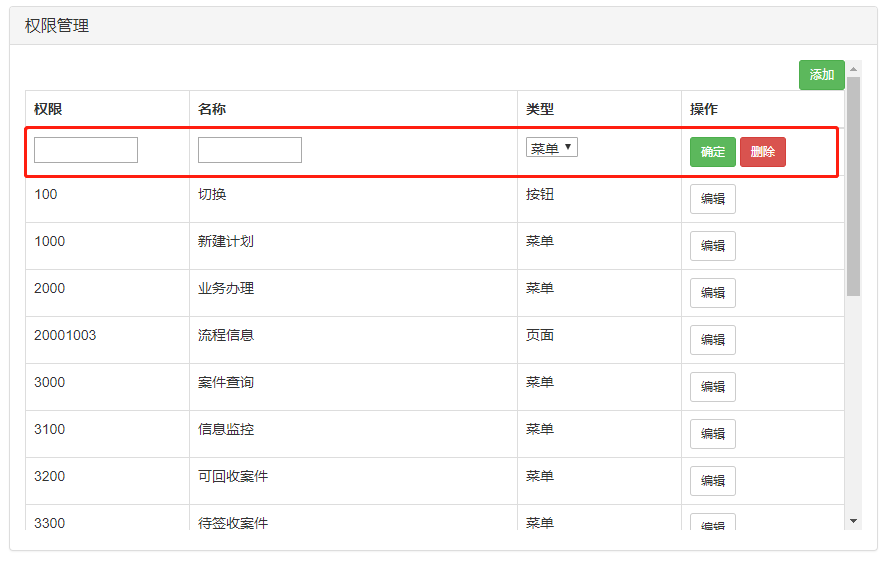
* 1. **权限管理**

权限管理界面，如图4-26所示，用户可以在该界面修改和添加角色。



**图4-26 权限管理界面**

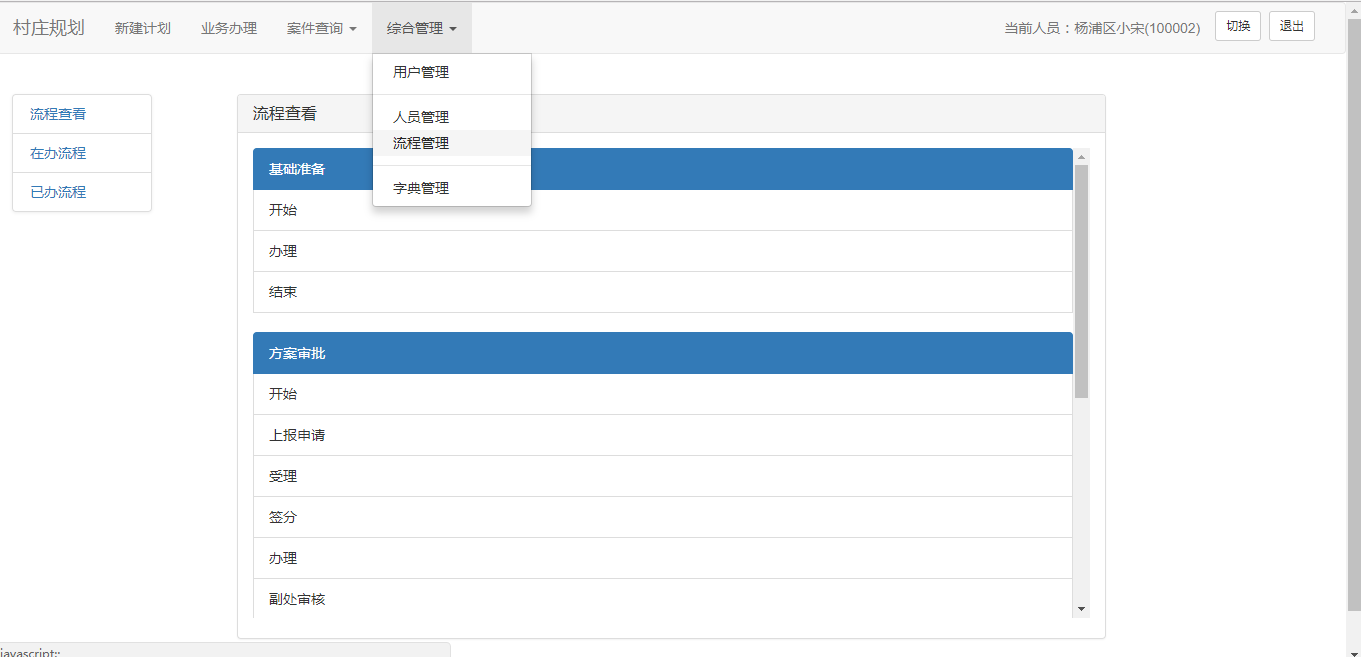
用户点击添加按钮之后，会要求用户填写新的权限对象信息，如图4-27所示。填写好后进行保存，即可使用新的权限对象跟角色进行关联操作。



**图4-27 添加权限**

* + 1. 工作流模块

流程管理模块中，除了以上提及的在案件详细信息界面中可以查看对应案件的流程信息，管理员还具有修改可调整流程的权限。如图4-28所示。



**图4-28 流程管理界面**

* + 1. **流程查看**

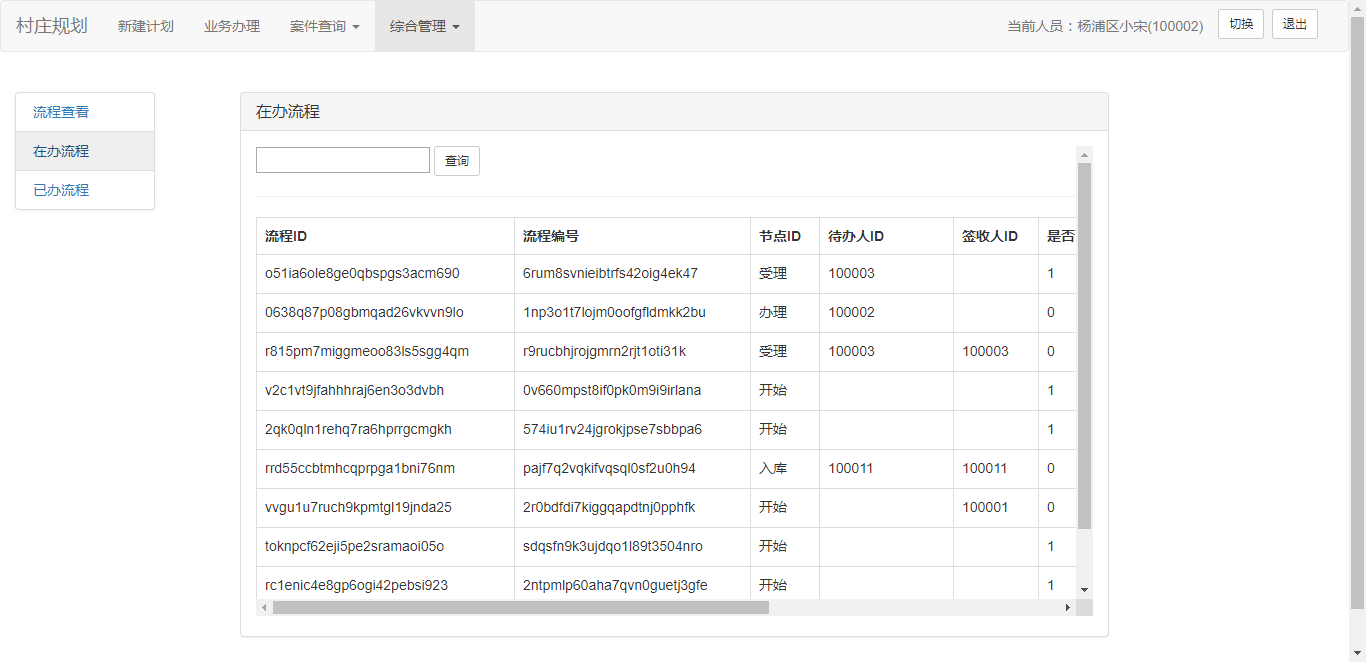
流程查看界面包含所有环节及环节下的节点信息，用户可以点击某一结点，展开该节点的详细信息，如图 4-29所示。

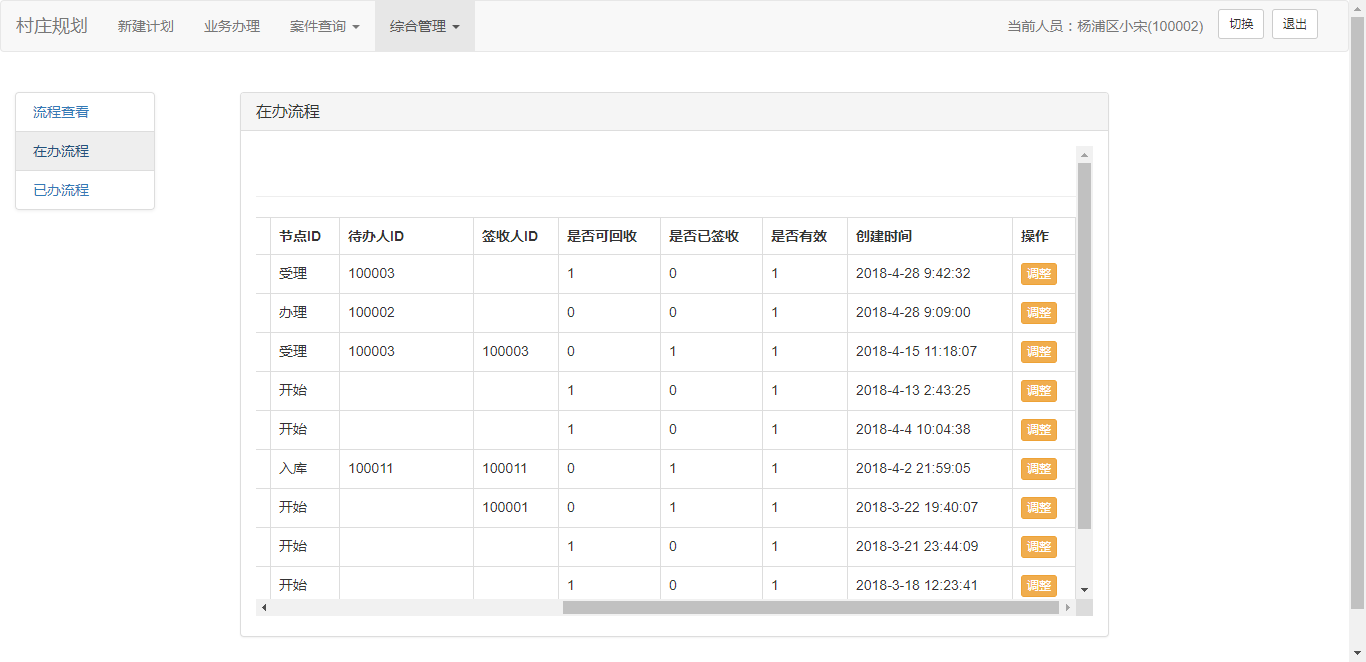


**图4-29 流程信息界面**

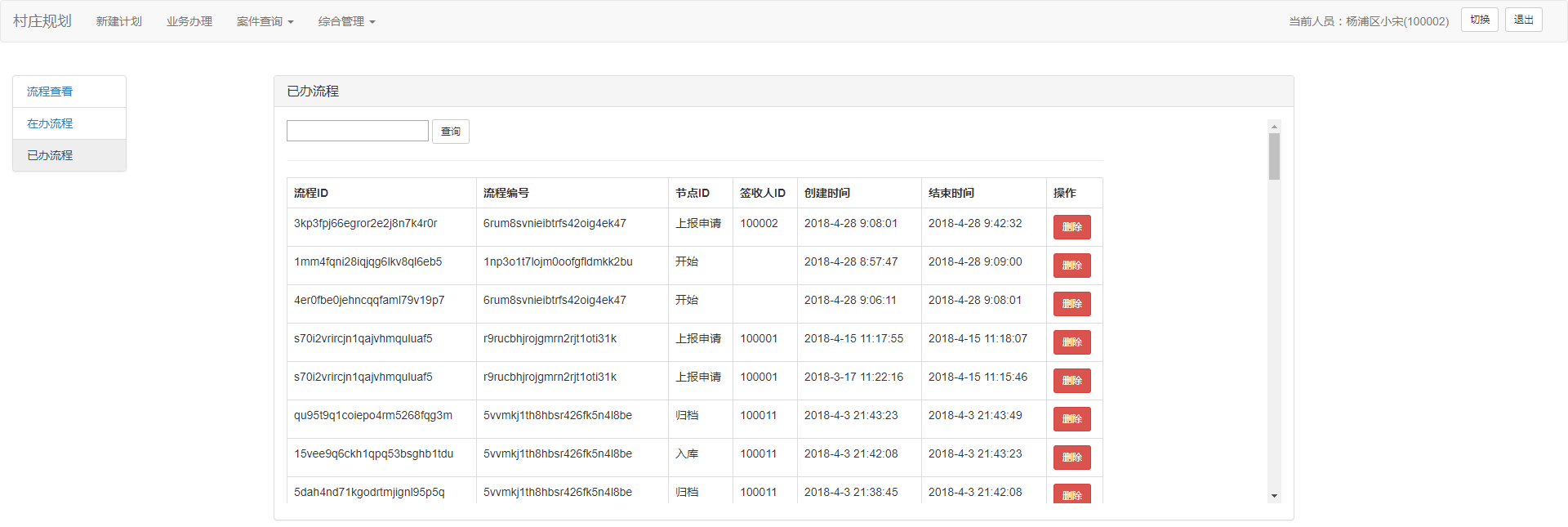
* + 1. **流程管理**

流程管理分为再版流程和已办流程两个部分。在办流程中，管理员可以查看所有流转中的流程实例，并可以对这些流程实例进行调整，如图4-30所示。已办流程中，用户可以根据流程编号查看已办流程，如图4-31所示。





**图4-30 在办流程**



**图4-31 已办流程**

* + 1. 其他

1. **字典**

系统上有对应的字段管理菜单，可以进入流程管理界面，如图4-32所示。左侧菜单分为三个主菜单，每个主菜单下又有相应的子菜单，点击子菜单即可显示相应的字典表记录。



**图4-32 字典管理界面**

1. **回收**

回收功能和签收功能有一定的逻辑关联性，能回收的案件是指用户进行提交操作后，下一界面的办理人没有签收，之前提交的办理可以将案件回收到自己的账号中，重新修改信息。回收界面如图4-33所示。



**图4-33 字典管理界面**

* 1. 详细设计
     1. 计划管理功能设计

1. 模块描述：对计划进行创建，删除，修改操作。
2. 代码构建集合

**表4-1计划管理代码构件集合**

|  |  |  |
| --- | --- | --- |
| 类名 | 包结构 | 职能 |
| PlanInfo.java | czsp.plan.model | 规划信息对象 |
| PlanApp.java | czsp.plan.model | 规划申请信息对象 |
| PlanInfoDao.java | czsp.plan.dao | 规划信息DAO层对象 |
| PlanAppDao.java | czsp.plan.dao | 规划申请信息DAO层对象 |
| PlanInfoService.java | czsp.plan.service | 规划业务层对象 |
| PlanModule.java | czsp.plan | 规划信息模块对象 |

1. 功能说明：

用户可以根据页面提示，点击相应的“新增计划”，“修改”，“删除”，“查询”按钮完成计划的增、改、删、查操作。如果填写的规划信息有误或后台报错，页面会提示相关错误信息。

* + 1. 计划启动功能设计

1. 模块描述：启动计划，用户能在方案审批中进行办理。
2. 代码构建集合

**表4-2计划启动代码构件集合**

|  |  |  |
| --- | --- | --- |
| 类名 | 包结构 | 职能 |
| PlanInfo.java | czsp.plan.model | 规划信息对象 |
| PlanApp.java | czsp.plan.model | 规划申请信息对象 |
| WfCurInstance.java | czsp.workflow.model | 当前流程对象 |
| WfRoute.java | czsp.workflow.model | 流程路由对象 |
| WfNode.java | czsp.workflow.model | 流程节点对象 |
| PlanInfoDao.java | czsp.plan.dao | 规划信息DAO层对象 |
| UserInfoDao.java | czsp.user.dao | 用户信息DAO层对象 |
| WFOperation.java | czsp.workflow | 流程操作类 |
| WfRouteDao.java | czsp.workflow.dao | 路由DAO层对象 |
| WfNodeDao.java | czsp.workflow.dao | 流程节点DAO层对象 |
| PlanInfoService.java | czsp.plan.service | 规划信息业务层对象 |
| PlanModule.java | czsp.plan | 规划信息模块对象 |

1. 功能说明：用创建计划时填写的环节id作为参数，根据路由表找到起始环节的其实节点，为计划创建相应的流程实例并启动，使案件进入流转状态，当前节点为选中的环节的起始节点。
   * 1. 案件审批功能设计
2. 模块描述：显示对应用户下待审批的案件列表。
3. 代码构建集合

**表4-3案件审批代码构件集合**

|  |  |  |
| --- | --- | --- |
| 类名 | 包结构 | 职能 |
| VplanWfDetail.java | czsp.plan.model.view | 案件详细信息视图（流程相关）对象 |
| UserInfo.java | czsp.user.model | 用户对象 |
| WfPhase.java | czsp.workflow.model | 流程环节对象 |
| WfPhaseDao.java | czsp.workflow.model | 流程路由DAO层对象 |
| WfDefineService.java | czsp.workflow.service | 流程定义业务层对象 |
| PlanInfoService.java | czsp.plan.service | 规划信息业务层对象 |
| PlanModule.java | czsp.plan | 规划信息模块对象 |

1. 功能说明：案件流转到某一待办人用户下，在案件审批菜单下的待办案件列表可以找到这类案件，用户可以对相关的案件进行签收和回收操作。
   * 1. 案件提交功能设计
2. 模块描述：推进案件的流程的信息，流转到下一节点。
3. 代码构建集合

**表4-4案件提交代码构件集合**

|  |  |  |
| --- | --- | --- |
| 类名 | 包结构 | 职能 |
| WfCurInstance.java | czsp.workflow.model | 当前流程对象 |
| VwfNodeDetail.java | czsp.workflow.model.view | 节点详细信息对象 |
| UserOperation.java | czsp.workflow.model | 用户操作对象 |
| WfHisInstance.java | czsp.workflow.model | 历史流程对象 |
| WfRoute.java | czsp.workflow.model | 流程路由对象 |
| planApp.java | czsp.plan.model | 规划申请信息对象 |
| PlanInfo.java | czsp.plan.model | 规划信息对象 |
| WfNode.java | czsp.workflow.model | 流程节点对象 |
| UserInfo.java | czsp.user.model | 用户对象 |
| WfRouteDao.java | czsp.workflow.dao | 路由DAO层对象 |
| WfNodeDao.java | czsp.workflow.dao | 流程节点DAO层对象 |
| UserOperationDao.java | czsp.user.dao | 用户操作DAO层对象 |
| UserInfoDao.java | czsp.user.dao | 用户信息DAO层对象 |
| PlanInfoDao.java | czsp.plan.dao | 规划信息DAO层对象 |
| PlanAppDao.java | czsp.plan.dao | 规划申请信息DAO层对象 |
| WfDefineService.java | czsp.workflow.service | 流程定义业务层对象 |
| UserOperationService.java | czsp.user.service | 用户操作业务层对象 |
| UserInfoService.java | czsp.user.service | 用户信息业务层对象 |
| WFOperation.java | czsp.workflow | 流程操作类 |
| WFModel.java | czsp.workflow | 流程模块 |

1. 功能说明：用户提交案件，修改流程表和app表以及主表中的信息，用户提交后，该案件会流转到相应人员的账号下，并由下一人员进行审批并提交。
   * 1. 意见填写功能设计
2. 模块描述：案件审批当中办理意见填写及查看。
3. 代码构建集合

**表4-5意见填写代码构件集合**

|  |  |  |
| --- | --- | --- |
| 类名 | 包结构 | 职能 |
| PlanOpinion.java | czsp.plan.model | 办理意见对象 |
| UserOperation.java | czsp.user.model | 用户操作对象 |
| PlanOpinionDao.java | czsp.user.model | 办理意见DAO层对象 |
| UserOperationDao.java | czsp.workflow.model | 用户操作DAO层对象 |
| PlanOpinionService.java | czsp.workflow.model | 用户操作业务层对象 |
| PlanModule.java | czsp.plan | 计划模块 |

1. 功能说明：用户可以填写、修改和保存当前节点的办理意见，系统会根据自己的筛选逻辑在流转的过程显示最新的用户办理意见。
   * 1. 案件查询功能设计
2. 模块描述：用户查询所有在系统创建的案件。
3. 代码构建集合

**表4-6案件查询代码构件集合**

|  |  |  |
| --- | --- | --- |
| 类名 | 包结构 | 职能 |
| VplanWfDetail.java | czsp.plan.model.view | 案件详细信息视图（流程相关）对象 |
| planInfo.java | czsp.plan.model | 规划信息对象 |
| PlanOpinion.java | czsp.plan.model | 办理意见对象 |
| PlanOpinionDao.java | czsp.plan.dao | 办理意见DAO层对象 |
| PlanInfoDao.java | czsp.plan.dao | 计划信息DAO层对象 |
| PlanInfoService.java | czsp.plan.service | 用户操作DAO层对象 |
| PlanOpinionService.java | czsp.plan.service | 办理意见DAO层对象 |
| PlanModule.java | czsp.plan | 计划模块 |

1. 功能说明：根据用户的查询条件查询在系统中的各个流转状态(未流转，流转中，办结)的案件，同时看以查看案件的详细信息。
   * 1. 权限管理功能设计
2. 模块描述：分配各个角色所能使用的系统功能。
3. 代码构建集合

**表4-7权限管理代码构件集合**

|  |  |  |
| --- | --- | --- |
| 类名 | 包结构 | 职能 |
| UserInfo.java | czsp.plan.model | 用户对象 |
| PermissionObject.java | czsp.authority.model | 权限对象 |
| PermissionRole.java | czsp.authority.model | 权限角色关联对象 |
| AccountInfo.java | czsp.account.model | 账号对象 |
| AccountUser.java | czsp.account.model | 账号用户关联对象 |
| UserInfoDao.java | czsp.user.dao | 用户DAO层对象 |
| PermissionRoleDao.java | czsp. authority.dao | 权限角色关联DAO层对象 |
| PermissionObjectDao.java | czsp.authority.dao | 权限DAO层对象 |
| AccountUserDao.java | czsp.account.dao | 账号用户关联DAO层对象 |
| AccountInfoDao.java | czsp.account.dao | 账号DAO层对象 |
| AccountService.java | czsp.account.service | 账号业务层对象 |
| PermissionService.java | czsp.authority.service | 权限业务层对象 |
| UserInfoService.java | czsp.user.service | 用户业务层对象 |
| AccountModule.java | czsp.account | 账户模块 |
| AuthorityModule.java | czsp.authority | 认证模块 |
| UserModule.java | czsp.user | 用户模块 |

1. 功能说明：创建权限，角色，用户，账户等对象，并将他们之间相互关联，从而实现对用户（账户）的权限管理，系统管理员具有操作各个角色的权限。
   1. 本章小结

本章通过系统操作截图展示了系统的主要功能，并从代码层划分了各功能的构建一览表，从概要设计到功能设计和详细设计，从各个角度介绍了村庄规划编审系统的计划管理，案件审批，查询统计，工作流，权限管理几个主要模块。详细设计中的模块介绍主要围绕大的模块功能进行说明，而进一步细分功能的实现，需要在后面的章节结合代码进行描述。

1. **上海市村庄规划编审平台数据库设计**
   1. 字典类数据表设计
      1. 总表

**表5-1 字典对照总表(BASE\_DIC)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **属性** | **类型** | **是否为空** | **主键** | **外键/对照表** | **说明** |
| DIC\_ID | VARCHAR2(100 BYTE) | No |  |  | 字典表id |
| TABLE\_NAME | VARCHAR2(100 BYTE) | Yes |  |  | 字典表名称 |

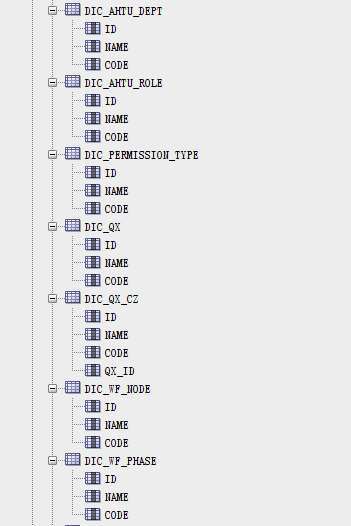
用户根据字典对照总表中的字典表id找到对应的字典表名称，并将字典表中的数据加载到系统缓存中，默认加载所有的字典表项。

* + 1. 查询分表

**表5-2 字典查询分表格式(DIC\_?)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **属性** | **类型** | **是否为空** | **主键** | **外键/对照表** | **说明** |
| ID | VARCHAR2(100 BYTE) | No | Yes |  | 字典项id |
| NAME | VARCHAR2(200 BYTE) | Yes |  |  | 字典项名称 |
| CODE | VARCHAR2(100 BYTE) | Yes |  |  | 字典项代码 |

图5-1展示了按照字典查询分表格式建表的各类型字典数据，包括：①部门、②角色、③权限类型、④区县、⑤村镇、⑥节点、⑦环节。



**图5-1 各类型字典表（对照列举）**

* 1. 业务类数据表设计
     1. 账号相关

**表5-3 账号信息表(ACCOUNT\_INFO)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **属性** | **类型** | **是否为空** | **主键** | **外键/对照表** | **说明** |
| USER\_NAME | VARCHAR2(100 BYTE) | No | Yes |  | 用户名 |
| PASSWORD | VARCHAR2(100 BYTE) | Yes |  |  | 密码 |
| IS\_AVAILABLE | CHAR(1 BYTE) | No |  |  | 是否可用 |
| UPDATE\_TIME | DATE | Yes |  |  | 更新时间 |
| CREATE\_TIME | DATE | Yes |  |  | 创建时间 |

账号信息表包括用户名、密码即可用标志位三个主要字段，主键为用户名，用户名的生成规则为根据用户姓名的拼音以及区县的代码进行组合，若用户为市局用户，则用户名不包括区县部分，如：杨浦区-张三对应yp\_zhangsan，市局-李四对应lisi。密码默认初始化六个1，对管理员提供修改所有用户密码的功能。

**表5-4 账号-用户关联表(ACCOUNT\_USER)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **属性** | **类型** | **是否为空** | **主键** | **外键/对照表** | **说明** |
| USER\_NAME | VARCHAR2(100 BYTE) | No | Yes | Yes/ACCOUNT\_INFO | 账户主键 |
| USER\_ID | VARCHAR2(100 BYTE) | No | Yes | Yes/USER\_INFO | 用户id |
| CREATE\_TIME | DATE | Yes |  |  | 关联时间 |
| IS\_DEFAULT | CHAR(1 BYTE) | Yes |  |  | 是否默认 |

帐号-用户关联表由复合主键构成，这两个键同时还是外键，关联着账号信息表中的用户名及用户信息表中的用户id，通常一个用户可以关联多个账号，并根据is\_default属性判断是否为默认绑定用户登录系统。

* + 1. 权限相关

**表5-5权限对象表(PERMISSION\_OBJECT)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **属性** | **类型** | **是否为空** | **主键** | **外键/对照表** | **说明** |
| O\_ID | VARCHAR2(100 BYTE) | No | Yes | Yes | 对象主键 |
| OBJECT\_TYPE | VARCHAR2(2 BYTE) | Yes |  |  | 1:菜单2.按钮3.url |
| OBJECT\_NAME | VARCHAR2(100 BYTE) | No |  |  | 对象名称 |
| CREATE\_TIME | DATE | Yes |  |  | 创建时间 |

权限对象表包括对象id，对象类型，对象名称三个主要字段，主键为对象id，生成规则由用户自定义，最大长度为100字节，对象类型根据说明，主要分为三类，存储的内容为其数字标号。

**表5-6 权限-角色关联表(PERMISSION\_ROLE)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **属性** | **类型** | **是否为空** | **主键** | **外键/对照表** | **说明** |
| ROLE\_ID | VARCHAR2(100 BYTE) | No | Yes | Yes/DIC\_AUTH\_ROLE | 对象主键 |
| OBJECT\_ID | VARCHAR2(100 BYTE) | No | Yes | Yes/PERMISSION\_OBJECT | 角色字典项id |

权限-角色关联表中存储着角色和权限的对应关系，它们之间是多对多的，不同用户可以具备相同的权限，同一个用户可具备多个权限。

* + 1. 规划相关

**表5-7 规划申请表(PLAN\_APP)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **属性** | **类型** | **是否为空** | **主键** | **外键/对照表** | **说明** |
| APP\_ID | VARCHAR2(100 BYTE) | No | Yes |  | 申请主键 |
| CUR\_NODE | VARCHAR2(10 BYTE) | Yes | Yes |  | 当前节点 |
| CUR\_PHASE | VARCHAR2(5 BYTE) | Yes |  |  | 当前环节 |
| INSTANCE\_NO | VARCHAR2(100 BYTE) | Yes |  |  | 实例编号 |
| LAST\_OP\_TIME | DATE | Yes |  |  | 最近操作时间 |
| LAST\_OP\_USER | VARCHAR2(100 BYTE) | Yes |  |  | 最近操作用户 |
| OPED\_USERS | VARCHAR2(2000 BYTE) | Yes |  |  | 操作过的用户 |
| PHASES | VARCHAR2(2000 BYTE) | Yes |  |  | 所选阶段 |
| STATUS | VARCHAR2(2 BYTE) | Yes |  |  | 状态：1.未流转，2.流转中，3.已办结 |

规划申请表中，主要存储着规划案件的操作信息和流程信息。其中主键app\_id的生成规则为32位的uuid，status存储的数据为流转状态前面的标号，操作过的用户和所选阶段分别为用户信息表中的多个user\_id以及流程环节表中的多个phase\_id，他们之间用逗号隔开。

**表5-8 规划信息表(PLAN\_INFO)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **属性** | **类型** | **是否为空** | **主键** | **外键/对照表** | **说明** |
| PLAN\_ID | VARCHAR2(100 BYTE) | No | Yes |  | 计划id |
| APP\_ID | VARCHAR2(100 BYTE) | Yes |  | Yes/PLAN\_INFO | 申请id |
| PLAN\_NAME | VARCHAR2(2000 BYTE) | Yes |  |  | 计划名称 |
| INSTANCE\_ID | VARCHAR2(100 BYTE) | Yes |  | Yes/WF\_CUR\_INSTANCE | 当前实例id |
| CREATE\_TIME | DATE | Yes |  |  | 创建时间 |
| CREATE\_USER\_ID | VARCHAR2(100 BYTE) | Yes |  |  | 创建人Id |
| IS\_FINISHED | VARCHAR2(1 BYTE) | Yes |  |  | 是否办结 |
| TOWN\_ID | VARCHAR2(100 BYTE) | Yes |  |  | 村镇id |
| TOWN\_NAME | VARCHAR2(200 BYTE) | Yes |  |  | 村镇名称 |
| PLAN\_AREA | NUMBER | Yes |  |  | 规划面积 |
| DESIGN\_DEPARTMENT | VARCHAR2(100 BYTE) | Yes |  |  | 设计单位 |
| DESIGN\_CONTACT\_NAME | VARCHAR2(100 BYTE) | Yes |  |  | 设计单位联系人 |
| DESIGN\_CONTACT\_WAY | VARCHAR2(100 BYTE) | Yes |  |  | 设计单位联系方式 |
| QX\_ID | VARCHAR2(100 BYTE) | Yes |  |  | 区县id |
| EXPECTED\_FINISH\_DATE | DATE | Yes |  |  | 预计办结日期 |
| FINISH\_DATE | DATE | Yes |  |  | 实际办结日期 |
| NOTE | VARCHAR2(4000 BYTE) | Yes |  |  | 备注 |

规划信息表为业务主表，里面包括着许多前端必填的业务字段（如，PLAN\_AREA、TOWN\_NAME等）以及一些重要的外键字段（如，APP\_ID、INSTANCE\_ID等），其中主键的PLAN\_ID的生成规则也为32位的uuid。

**表5-9 办理意见表(PLAN\_OPINION)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **属性** | **类型** | **是否为空** | **主键** | **外键/对照表** | **说明** |
| NODE\_ID | VARCHAR2(100 BYTE) | Yes |  | Yes/WF\_NODE | 节点id |
| CREATE\_BY | VARCHAR2(100 BYTE) | Yes |  | Yes/USER\_INFO | 创建人id |
| CREATE\_TIME | DATE | Yes |  |  | 创建时间 |
| OPINION\_CONTENT | VARCHAR2(4000 BYTE) | Yes |  |  | 意见内容 |
| PLAN\_ID | VARCHAR2(100 BYTE) | Yes |  | Yes/PLAN\_INFO | 案件id |
| OP\_TYPE | VARCHAR2(100 BYTE) | Yes |  |  | 操作类型 |
| PHASE\_ID | VARCHAR2(100 BYTE) | Yes |  | Yes/WF\_PHASE | 环节id |
| INSTANCE\_ID | VARCHAR2(100 BYTE) | Yes |  | Yes/WF\_CUR\_INSTANCE | 当前节点id |
| UPDATE\_TIME | DATE | Yes |  |  | 更新时间 |
| OPINION\_ID | VARCHAR2(100 BYTE) | No | Yes |  | 意见id |
| CREATE\_USER\_NAME | VARCHAR2(100 BYTE) | Yes |  |  | 创建人 |

办理意见表中主键为OPINION\_ID,生成规则为 32位的uuid。其中每条记录绑定多项外键：包括用户，环节，流程，规划信息，节点等。这样在筛选时可以做到更好的过滤和更精确的选择。

* + 1. 用户相关

**表5-10 用户信息表(USER\_INFO)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **属性** | **类型** | **是否为空** | **主键** | **外键/对照表** | **说明** |
| USER\_ID | VARCHAR2(100 BYTE) | No | Yes |  | 用户id |
| NAME | VARCHAR2(100 BYTE) | Yes |  |  | 姓名 |
| DEPARTMENT\_ID | VARCHAR2(100 BYTE) | Yes |  | Yes/DIC\_AUTH\_DEPARTMENT | 部门id |
| ROLE\_ID | VARCHAR2(200 BYTE) | Yes |  | Yes/DIC\_AUTH\_ROLE | 角色id |
| QX\_ID | VARCHAR2(100 BYTE) | Yes |  | Yes/DIC\_QX | 区县id |
| PHONE\_NUMBER | VARCHAR2(100 BYTE) | Yes |  |  | 电话号码 |
| SEX | CHAR(1 BYTE) | Yes |  |  | 性别：0男1女 |

用户信息表中，用户的主键生成规则为触发器和对应序列赋值，关于触发器的序列说明参照后面编号5.4其他中的内容，用户信息包括基本属性（sex，phone\_number，name）以及相关的字典表外键（department\_id，role\_id，qx\_id）。

**表5-11 用户操作记录表(USER\_OPERATION)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **属性** | **类型** | **是否为空** | **主键** | **外键/对照表** | **说明** |
| OP\_TYPE | VARCHAR2(100 BYTE) | No |  |  | 操作类型 |
| OP\_CREATE\_TIME | DATE | Yes |  |  | 操作时间 |
| NODE\_ID | VARCHAR2(100 BYTE) | Yes |  |  | 节点id |
| USER\_ID | VARCHAR2(100 BYTE) | Yes |  | Yes/USER\_INFO | 用户id |
| INSTANCE\_ID | VARCHAR2(100 BYTE) | Yes |  |  | 操作后流程id |
| PRE\_INSTANCE\_ID | VARCHAR2(100 BYTE) | Yes |  |  | 操作前流程id |

用户操作记录表中包括instance\_id、pre\_instance\_id两个关键属性，这两个属性引导系统查询流程实例的生成情况并按流程id展示交替情况。op\_type字段说明用户的操作类型。

* + 1. 流程相关

**表5-12 当前流程表(WF\_CUR\_INSTANCE)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **属性** | **类型** | **是否为空** | **主键** | **外键/对照表** | **说明** |
| INSTANCE\_ID | VARCHAR2(100 BYTE) | No | Yes |  | 流程主键 |
| INSTANCE\_NO | VARCHAR2(100 BYTE) | Yes |  |  | 流程编号 |
| NODE\_ID | VARCHAR2(100 BYTE) | No |  | Yes/WF\_NODE | 节点编号 |
| IF\_SIGN | VARCHAR2(1 BYTE) | Yes |  |  | 是否签收 |
| IF\_RETRIEVE | VARCHAR2(1 BYTE) | Yes |  |  | 是否可回收 |
| IF\_VALID | VARCHAR2(1 BYTE) | Yes |  |  | 是否有效 |
| CREATE\_TIME | DATE | Yes |  |  | 创建时间 |
| SIGN\_USER\_ID | VARCHAR2(100 BYTE) | Yes |  | Yes/USER\_INFO | 签收用户id |
| TODO\_USER\_ID | VARCHAR2(100 BYTE) | Yes |  |  | 待办用户id |

当前流程表中的主键为instance\_id，生成规则为32位的uuid，流程编号instance\_no生成规则也为32位的uuid，一个流程id一般对应一个可操作节点，一个流程编号一般对应一个流程环节。if\_sign，if\_retrieve，if\_valid是三个业务逻辑判断的标志位。todo\_user\_id对应一个或多个user\_id，如果对应多个user\_id则用逗号分隔。

**表5-13 历史流程表(WF\_HIS\_INSTANCE)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **属性** | **类型** | **是否为空** | **主键** | **外键/对照表** | **说明** |
| INSTANCE\_ID | VARCHAR2(100 BYTE) | Yes |  |  | id |
| INSTANCE\_NO | VARCHAR2(100 BYTE) | Yes |  |  | 编号 |
| NODE\_ID | VARCHAR2(100 BYTE) | Yes |  | Yes/WF\_NODE | 节点id |
| SIGN\_USER\_ID | VARCHAR2(100 BYTE) | Yes |  | Yes/USER\_INFO | 签收用户id |
| CREATE\_TIME | DATE | Yes |  |  | 创建时间 |
| FINISH\_TIME | DATE | Yes |  |  | 结束时间 |

历史流程表中的信息基本是从当前流程表中归档而来的，基本上是复制处理完的当前流程的id，除了这些从当前流程表复制的属性，还有一个finish\_time的属性记录着处理完成的时间（同时也是历史流程表创建新纪录的时间）。

**表5-14 流程节点表(WF\_NODE)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **属性** | **类型** | **是否为空** | **主键** | **外键/对照表** | **说明** |
| NODE\_ID | VARCHAR2(100 BYTE) | No | Yes |  | 节点id |
| NODE\_NAME | VARCHAR2(200 BYTE) | Yes |  |  | 节点名称 |
| PHASE\_ID | VARCHAR2(100 BYTE) | Yes |  |  | 阶段id |
| ROLE\_ID | VARCHAR2(100 BYTE) | Yes |  |  | 角色id |
| WF\_CUR\_NODE | VARCHAR2(2 BYTE) | Yes |  |  | 当前节点编号 |
| IS\_END | VARCHAR2(1 BYTE) | Yes |  |  | 是否是尾节点 |
| IS\_START | VARCHAR2(1 BYTE) | Yes |  |  | 头节点 |
| IS\_QX\_OP | VARCHAR2(1 BYTE) | Yes |  |  | 是否区县操作节点 |

流程节点表中主键为node\_id，生成规则参考对应节点字典表的主键。is\_start和is\_end为判断属性，通过该判断属性可以找到某一环节的初始节点和结束节点。role\_id决定了哪些角色具有操作此节点的权限，可对应多个角色，用逗号分隔，is\_qx\_op判断此节点是否为区县操作节点，如果为区县操作节点，则在选择经办人时，角色筛选下再加上用户区县筛选。

**表5-15 流程环节表(WF\_PHASE)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **属性** | **类型** | **是否为空** | **主键** | **外键/对照表** | **说明** |
| PHASE\_ID | VARCHAR2(100 BYTE) | No | Yes |  | 环节id |
| PHASE\_NAME | VARCHAR2(200 BYTE) | Yes |  |  | 环节名称 |
| PRE\_PHASE\_ID | VARCHAR2(100 BYTE) | Yes |  |  | 前置环节 |
| NEXT\_PHASE\_ID | VARCHAR2(100 BYTE) | Yes |  |  | 下一环节 |
| IS\_END | VARCHAR2(1 BYTE) | Yes |  |  | 尾 |
| IS\_START | VARCHAR2(1 BYTE) | Yes |  |  | 头 |
| WF\_CODE | VARCHAR2(100 BYTE) | Yes |  |  | 环节代码 |

流程环节表中主键为phase\_id，生成规则参考对应环节字典表的主键。is\_start和is\_end为判断属性，通过该判断属性可以找到总流程的初始环节和结束环节。pre\_phase\_id和next\_phase\_id指向前后环节的环节主键。

**表5-16 流程路由表(WF\_ROUTE)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **属性** | **类型** | **是否为空** | **主键** | **外键/对照表** | **说明** |
| ROUTE\_ID | VARCHAR2(100 BYTE) | No | Yes |  | 路由主键 |
| PHASE\_ID | VARCHAR2(100 BYTE) | Yes |  | Yes/WF\_PHASE | 环节id |
| CUR\_NODE | VARCHAR2(2 BYTE) | Yes |  |  | 当前 |
| NEXT\_NODE | VARCHAR2(2 BYTE) | Yes |  |  | 后置 |
| PRE\_NODE | VARCHAR2(100 BYTE) | Yes |  |  | 前驱 |
| IS\_TESONG | VARCHAR2(1 BYTE) | Yes |  |  | 是否特送 |
| DISPLAY\_ORDER | VARCHAR2(2 BYTE) | Yes |  |  | 显示排序 |

流程路由表主要帮助展示环节内节点之间的拓扑图，该图为有向图，主键route\_id的生成规则为环节id+源节点id+目的节点id，若特送标志位is\_tesong的值为空，说明下一节点next\_node为默认的指向节点。属性display\_order帮助页面控制节点选择的显示顺序。

* 1. 业务类视图设计

**表5-17 规划详细信息视图/查询相关(V\_PLAN\_INFO\_DETAIL)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **序号** | **属性** | **类型** | **是否为空** | **来源** |
| 1 | APP\_ID | VARCHAR2(100) | Yes | PLAN\_INFO |
| 2 | CREATE\_TIME | DATE | Yes | PLAN\_INFO |
| 3 | CREATE\_USER\_ID | VARCHAR2(100) | Yes | PLAN\_INFO |
| 4 | INSTANCE\_ID | VARCHAR2(100) | Yes | PLAN\_INFO |
| 5 | PLAN\_ID | VARCHAR2(100) | No | PLAN\_INFO |
| 6 | PLAN\_NAME | VARCHAR2(2000) | Yes | PLAN\_INFO |
| 7 | TOWN\_NAME | VARCHAR2(200) | Yes | PLAN\_INFO |
| 8 | QX\_ID | VARCHAR2(100) | Yes | PLAN\_INFO |
| 9 | PLAN\_AREA | NUMBER | Yes | PLAN\_INFO |
| 10 | CREATE\_USER\_NAME | VARCHAR2(100) | Yes | USER\_INFO |
| 11 | CUR\_PHASE | VARCHAR2(5) | Yes | PLAN\_APP |
| 12 | CUR\_NODE | VARCHAR2(10) | Yes | PLAN\_APP |
| 13 | STATUS | VARCHAR2(2) | Yes | PLAN\_APP |

该视图从PLAN\_INFO，USER\_INFO，PLAN\_APP三张表中筛选字段，更方便地展示计划列表界面

**表5-18规划详细信息视图/流程相关(V\_PLAN\_WF\_DETAIL)**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **序号** | **属性** | | **类型** | | **是否为空** | | **来源** | |
| 1 | PLAN\_ID | | VARCHAR2(100) | | No | | PLAN\_INFO | |
| 2 | PLAN\_NAME | | VARCHAR2(2000) | | Yes | | PLAN\_INFO | |
| 3 | INSTANCE\_ID | | VARCHAR2(100) | | Yes | | PLAN\_INFO | |
| 4 | QX\_ID | | VARCHAR2(100) | | Yes | | PLAN\_INFO | |
| 5 | CREATE\_TIME | | DATE | | Yes | | PLAN\_INFO | |
| 6 | TOWN\_NAME | | VARCHAR2(200) | | Yes | | PLAN\_INFO | |
| 7 | TODO\_USER\_ID | | VARCHAR2(100) | | Yes | | WF\_CUR\_INSTANCE | |
| 8 | SIGN\_USER\_ID | | VARCHAR2(100) | | Yes | | WF\_CUR\_INSTANCE | |
| 9 | IF\_SIGN | | VARCHAR2(1) | | Yes | | WF\_CUR\_INSTANCE | |
| 10 | IF\_RETRIEVE | | VARCHAR2(1) | | Yes | | WF\_CUR\_INSTANCE | |
| 11 | SIGN\_USER\_NAME | | VARCHAR2(100) | | Yes | | USER\_INFO | |
| 12 | APP\_ID | | VARCHAR2(100) | | Yes | | PLAN\_APP | |
| 13 | CUR\_NODE | | VARCHAR2(10) | | Yes | | PLAN\_APP | |
| 14 | CUR\_PHASE | | VARCHAR2(5) | | Yes | | PLAN\_APP | |
| 15 | INSTANCE\_NO | | VARCHAR2(100) | | Yes | | PLAN\_APP | |
| 16 | LAST\_OP\_TIME | | DATE | | Yes | | PLAN\_APP | |
| 17 | LAST\_OP\_USER | | VARCHAR2(100) | | Yes | | PLAN\_APP | |
| 18 | OPED\_USERS | | VARCHAR2(2000) | | Yes | | PLAN\_APP | |
| 19 | PHASES | | VARCHAR2(2000) | | Yes | | PLAN\_APP | |
| 20 | STATUS | | VARCHAR2(2) | | Yes | | PLAN\_APP | |
| 13 | | STATUS | | VARCHAR2(2) | | Yes | | PLAN\_APP |

该视图从PLAN\_INFO，USER\_INFO，PLAN\_APP，WF\_CUR\_INSTANCE四张表中筛选字段，更方便地展示案件审批列表，案件查询，签收回收列表界面。

**表5-17 节点详细信息(V\_WF\_NODE\_DETAIL)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **序号** | **属性** | **类型** | **是否为空** | **来源** |
| 1 | NODE\_ID | VARCHAR2(100) | No | WF\_NODE |
| 2 | NODE\_NAME | VARCHAR2(200) | Yes | WF\_NODE |
| 3 | WF\_CUR\_NODE | VARCHAR2(2) | Yes | WF\_NODE |
| 4 | IS\_START | VARCHAR2(1) | Yes | WF\_NODE |
| 5 | IS\_END | VARCHAR2(1) | Yes | WF\_NODE |
| 6 | PHASE\_ID | VARCHAR2(100) | Yes | WF\_PHASE |
| 7 | PHASE\_NAME | VARCHAR2(200) | Yes | WF\_PHASE |
| 8 | WF\_CODE | VARCHAR2(100) | Yes | WF\_PHASE |

该视图连接了节点表WF\_NODE和环节表WF\_PHASE,辅助执行根据路由查询所有到达所有节点的业务逻辑。

* 1. 其他

1. 触发器

create or replace TRIGGER TRI\_ADD\_USER

BEFORE INSERT ON USER\_INFO

for each row

BEGIN

select seq\_user\_id.nextval into :new.user\_id from dual;

END;

这段sql代码声明了一个触发器，当添加用户信息（USER\_INFO表）时，取到seq\_user\_id序列中的下一值并赋给这条新增记录的主键。

1. 序列

CREATE SEQUENCE "CZSP"."SEQ\_USER\_ID" MINVALUE 100001 MAXVALUE 999999 INCREMENT BY 1 START WITH 100121 CACHE 20 NOORDER NOCYCLE ;

这段sql代码声明了一个序列，从100001开始递增，每次递增的值为1，递增后的值将作为对应TRI\_ADD\_USER触发器中所取的值。

1. **上海市村庄规划编审平台系统实现**
   1. 配置文件
      1. web.xml

……

<filter>

<filter-name>nutz</filter-name>

<filter-class>org.nutz.mvc.NutFilter</filter-class>

<init-param>

<param-name>modules</param-name>

<param-value>czsp.MainModule</param-value>

</init-param>

<init-param>

<param-name>exclusions</param-name>

<param-value>/static/\*</param-value>

</init-param>

</filter>

<filter-mapping>

<filter-name>nutz</filter-name>

<url-pattern>/\*</url-pattern>

<dispatcher>REQUEST</dispatcher>

<dispatcher>FORWARD</dispatcher>

</filter-mapping>

<welcome-file-list>

<welcome-file>/czsp/login.jsp</welcome-file>

</welcome-file-list>

</web-app>

在web.xml中配置nutz的过滤器org.nutz.mvc.NutFilter，除去站点名称之后的url地址全都经过过滤器找到对应的入口。同时指定加载nutz框架时的启动类，声明了哪些路径下的文件不用经过过滤即可访问，指定默认欢迎页面(登录页面)。

* 1. 实体映射

关于实体映射，也采用nutz框架下的org.nutz.dao.entity.annotation包中的注解实现，如以下代码：

**@Table**("PLAN\_INFO")

public class PlanInfo {

**@Name**

**@Prev**(els = **@EL**("uuid(32)"))

**@Column**(value = "PLAN\_ID")

private String planId;

@Prev(els = @EL("uuid(32)"))

@Column(value = "APP\_ID")

private String appId;

**@One**(field = "appId")

private PlanApp planApp;

@Column(value = "PLAN\_NAME")

private String planName;

……

public String getPlanId() {

return planId;

}

public void setPlanId(String planId) {

this.planId = planId;

}

public String getAppId() {

return appId;

}

public void setAppId(String appId) {

this.appId = appId;

}

public String getPlanName() {

return planName;

}

public void setPlanName(String planName) {

this.planName = planName;

}

}

1. @Table：声明的一个 POJO 所对应的数据表名(@View同理)。
2. @Name：声明该字段对应数据库字符型主键。
3. @Prev：在插入一条记录前，对某个字段进行赋值，本处使用nutz内置uuid方法生成id并赋值。
4. @EL：输入的参数为一串el表达式代码。
5. @One：在一个字段上声明一条一对一映射，这个声明需要输入两个参数，一个是这个字段对应的实体类，本处为PlanApp；另一个是关联的字段名，此处是appId。
   1. 代码层模块
      1. 主模块

@Modules(scanPackage = true)

@SetupBy(MainSetup.class)

@IocBy(args={"\*org.nutz.ioc.loader.json.JsonLoader", "ioc/dao.js",

"\*org.nutz.ioc.loader.annotation.AnnotationIocLoader","czsp"

})

public class MainModule {

}

1. @Modules中声明自动扫描主模块所在包下的所有类，如果某个类中有两个以上的入口函数，则会判定成模块类。
2. @SetupBy中声明了在应用启动时需要执行的字节码文件，这些是内容是除了加载nutz过滤器和注册bean之类默认内容的额外内容，此处的MainSetup代码如下所示:

public class MainSetup implements Setup{

@Override

public void init(NutConfig nc) {

Ioc ioc = nc.getIoc(); // 拿到Ioc容器

Dao dao = ioc.get(Dao.class); // 通过Ioc容器可以拿到你想要的ioc bean

final Log log = Logs.getLog(MainSetup.class);

// 初始化字典工具

DicUtil.getInstance();

// 加载所有字典表名称

List<Record> records = dao.query("base\_dic", null);

for (Record record : records) {

String tableName = record.get("TABLE\_NAME").toString();

//加载字典表记录

List<Record> dicTable = dao.query(tableName, null);

Map dicMap = new TreeMap();

for(Record dicItem : dicTable){

dicMap.put(dicItem.get("ID"), dicItem);

}

DicUtil.getInstance().addDic(record.get("DIC\_ID").toString(), dicMap);

log.info("already add dictable:"+tableName);

}

}

@Override

public void destroy(NutConfig nc) {

// TODO Auto-generated method stub

}

}

其中init()方法为应用启动时的方法，destroy()方法为应用关闭时的方法。此处仅在启动时添加额外的操作步骤：

* 1. 获得ioc容器并从容器中拿到数据库访问对象dao。
  2. 加载日志。
  3. 初始化字典，相关代码请查看本块6.4.1部分。
  4. 从字典对照总表中读出所有的字典查询分表信息
  5. 根据字典分表信息的分表名称，加载分表名称中所有的记录，并放到系统缓存中，以便读取。

1. @IocBy生命了哪些对象是要根据配置文件初始化并由容器管理的，此处的加载方式包括两种，一个是JsonLoader根据json字符串中的内容加载对象，内容如下：

var ioc = {

/\*

\* 数据库连接池

\*/

dataSource : {

type : "com.mchange.v2.c3p0.ComboPooledDataSource",

fields : {

driverClass : "oracle.jdbc.driver.OracleDriver",

jdbcUrl : "jdbc:oracle:thin:@localhost:1521:orcl",

user : "czsp",

password : "czsp"

}

},

/\*

\* 这个配置很好理解， args 表示这个对象构造函数的参数。显然，下面的注入方式将调用 new NutDao(dataSource)

\*/

dao : {

type : "org.nutz.dao.impl.NutDao",

args : [ {

refer : "dataSource"

} ]

},

};

加载bean时参考以上MainSetup类中的应用启动时方法的第一步骤代码即可完成变量初始化。

除此之外，另一种方式是基于注解的bean加载，在需要注册的类前加上@IocBean注解，调用时在对应的变量前添加@Inject即可完成属性注入。如以下两部分代码所示：

@IocBean

public class UserInfoDao {

……

}

@IocBean

public class UserInfoService {

**@Inject**

**private UserInfoDao userInfoDao;**

@Inject

private PermissionRoleDao permissionRoleDao;

……

}

* + 1. 子模块
       1. Dao层

Dao全称为Data Access Object，即数据库访问对象，为了降低耦合度，一般一实体类对应一个Dao类。Dao类中一般包含数据库对象的增删改查方法。

本平台用的是nutz框架中的Dao类，它是DML操作的核心接口，里面封装了大量的数据库操作。基本的数据操作代码如下所示：

@IocBean

public class PlanInfoDao {

**Ioc ioc = Mvcs.getIoc();**

**Dao dao = ioc.get(Dao.class, "dao");**

public PlanInfo add(PlanInfo planInfo) {

planInfo.setCreateTime(new Date());

return **dao.insert(planInfo);**

}

public void deletePlan(String planId) {

**dao.delete(PlanInfo.class, planId);**

}

public void update(PlanInfo planInfo) {

**dao.update(planInfo);**

}

public List getListByCondition(VplanWfDetail planCondition, Object orderBy) {

Criteria cri = Cnd.cri();

// 待办人id和签收人id

if (StringUtils.isNotBlank(planCondition.getTodoUserId())

&& StringUtils.isNotBlank(planCondition.getSignUserId())) {

SqlExpressionGroup e1 = Cnd.exps("todoUserId", "LIKE", "%" + planCondition.getTodoUserId() + "%")

.and("ifSign", "=", "0");

SqlExpressionGroup e2 = Cnd.exps("signUserId", "=", planCondition.getSignUserId()).and("ifSign", "=", "1");

cri.where().or(e1).or(e2);

}

// 状态

if (StringUtils.isNotBlank(planCondition.getStatus()))

cri.where().andEquals("status", planCondition.getStatus());

if (orderBy == null)

cri.getOrderBy().desc("create\_time");

return **dao.query(VplanWfDetail.class, cri);**

}

}

首先在dao类中声明字段ioc并通过Mvcs类得到环境中的ioc容器，再通过从容器中取bean初始化dao变量。接着可以用成员方法insert(T t)、delete(String id)、update(Object o)、query(Class classOfT,Condition cnd)实现对数据库的增删改查操作。由于在定义实体类的时候，已经给对应的字段加上了nutz实体注解，所以不用做额外的映射操作。

* + - 1. Service层

@IocBean

public class PlanInfoService {

@Inject

private PlanInfoDao planInfoDao;

@Inject

private PlanAppDao planAppDao;

@Inject

private WFOperation wfOperation;

public void add(PlanInfo newPlan, PlanApp newPlanApp) {

**Trans.exec(new Atom() {**

**public void run() {**

// 初始化info表

newPlan.setIsFinished("0");

PlanInfo planInfo = planInfoDao.add(newPlan);

// 初始化app表

PlanApp planApp = planAppDao.add(planInfo, newPlanApp);

// 初始化流程表

WfCurInstance curInfstance = wfOperation.initInstance(planApp.getCurPhase());

// 更新info表关联字段

planInfo.setInstanceId(curInfstance.getInstanceId());

planInfoDao.update(planInfo);

// 更新app表关联字段

planApp.setInstanceNo(curInfstance.getInstanceNo());

planApp.setCurNode(curInfstance.getNodeId());

planAppDao.update(planApp);

**}**

**});**

}

……

}

一块业务逻辑通常对应一个Service类，一个service类下可能需要协同多个dao类完成操作，以上的PlanSerice类中就包括三个dao字段。根据业务逻辑分别对三张数据库表进行操作。为了保证各个操作的原子性，可以使用nutz框架的org.nutz.trans包下的Trans和Atom类，通过内置的事务模板(如以上代码所示)实现包裹代码中的所有dao操作都是原子性的。

* + - 1. Module层

@IocBean

**@At**("/plan")

public class PlanModule {

@Inject

private PlanInfoService planInfoService;

**@At**("/edit/?")

**@Ok**("jsp:/czsp/plan/edit")

public Map<String, Object> edit(String planId) {

Map<String, Object> map = new HashMap<String, Object>();

PlanInfo planInfo = planInfoService.getPlanInfoByPlanId(planId);

Map dicQx = (TreeMap) (DicUtil.getInstance().getDicMap().get(Constants.DIC\_QX\_NO));

map.put("planInfo", planInfo);

map.put("dicQx", dicQx);

return map;

}

**@At**("/update")

**@AdaptBy**(type = JsonAdaptor.class)

**@Ok**("json")

public Map<String, Object> update(@Param("..") PlanInfo planInfo) {

Map<String, Object> map = new HashMap<String, Object>();

try {

planInfoService.update(planInfo);

map.put("result", "success");

} catch (Exception e) {

map.put("result", "fail");

map.put("message", MessageUtil.getStackTraceInfo(e));

}

return map;

}

}

Module类为模块类，与实际的功能模块最为匹配。@At注解可以声明在模块类上，也可以模块类中的入口函数上，如果声明在模块类上，访问的路径是由声明在模块类上的注解的参数地址加上入口函数上@At注解中参数地址组成的。

以上代码中主要展示了，两种常用的数据接收方式，和返回的数据类型。其中第二个方法是通过声明@AdaptBy注解，并制定解析参数方式为json，可将前台的json字符串解析成对应的实体对象，同时在执行完方法之内的代码之后，根据@Ok注解中的参数指定，向客户端返回的数据也为json格式。第一个方法接受的客户端不需要经过json适配器处理即可匹配到入口函数的方法中，同时返回的内容部位json数据，而是对应@Ok内参数指明路径下的jsp页面，在页面上需要的参数通过取函数的返回值Map<String, Object>中的数据进行渲染。

* 1. 核心业务逻辑
  2. 其他
     1. 工具类
     2. 常量