**《快速开发平台》**

**设计使用说明文档**

|  |  |  |  |
| --- | --- | --- | --- |
| **公司名称** | 江苏博达集优智能科技股份有限公司 | **文档编号** | 001 |
| **文档名称** | 设计使用说明书 | **起草日期** | 2015-06-13 |
| **起草** | 朱允伟 | **修改日期** | 2015-07-08 |

**目录**

[1 概述 4](#_Toc422210455)

[1.1 目的和范围 4](#_Toc422210456)

[1.2 平台基本架构： 4](#_Toc422210457)

[1.3 客户端技术 5](#_Toc422210458)

[1.4 服务端技术 6](#_Toc422210459)

[1.5 设计器及模型表示 7](#_Toc422210460)

[1.6 组织建模与权限控制 9](#_Toc422210461)

[1.7 工作流系统 10](#_Toc422210462)

[1.8 术语和定义 11](#_Toc422210463)

[1.9 参考资料 12](#_Toc422210464)

[1.10 功能划分 12](#_Toc422210465)

[1.11 常规界面规范 12](#_Toc422210466)

[1.12 设计规范 13](#_Toc422210467)

[2 应用程序体系设计 14](#_Toc422210468)

[2.1 简述 14](#_Toc422210469)

[2.2 设计器主界面 14](#_Toc422210470)

[2.3 应用程维护 15](#_Toc422210471)

[2.4 应用程序模块维护 15](#_Toc422210472)

[2.5 应用程序模块关联 16](#_Toc422210473)

[2.6 命名空间维护 16](#_Toc422210474)

[2.7 打开当前应用程序 17](#_Toc422210475)

[2.8 类型维护 18](#_Toc422210476)

[2.8.1 实体类型维护 18](#_Toc422210477)

[2.8.2 关联类型维护 26](#_Toc422210478)

[2.9 控制类维护 26](#_Toc422210479)

[2.10 工作流设计器 28](#_Toc422210480)

[3 界面设计 29](#_Toc422210481)

[3.1 简述 29](#_Toc422210482)

[3.1.1 字典页面模型设计 29](#_Toc422210483)

[3.1.2 单据列表页面模型设计 32](#_Toc422210484)

[3.1.3 单据维护页面模型设计 34](#_Toc422210485)

[3.1.4 树结构页面模型设计 36](#_Toc422210486)

# 概述

## 目的和范围

信息系统的开发中，用户的需求和最终的实现往往跨越了一个巨大的鸿沟，需求变成实现往往要经过一个复杂的设计、开发、调试、发布、试用、调整等过程，看似简单的一些需求和变更，都会需要很大的工作量才能完成。出现这种情况的最主要原因是用户需求和功能实现之间的失谐，如果能取消设计和实现之间的多个阶段，即可极大的减少中间步骤的工作量，实现系统的快速实现、快速变更能力。

设计一个程序使得客户或者二次开发人员可以使用程序自主设计类来实现其功能

统一平台关注以下几个要素：

领域实体的动态配置和服务的动态管理

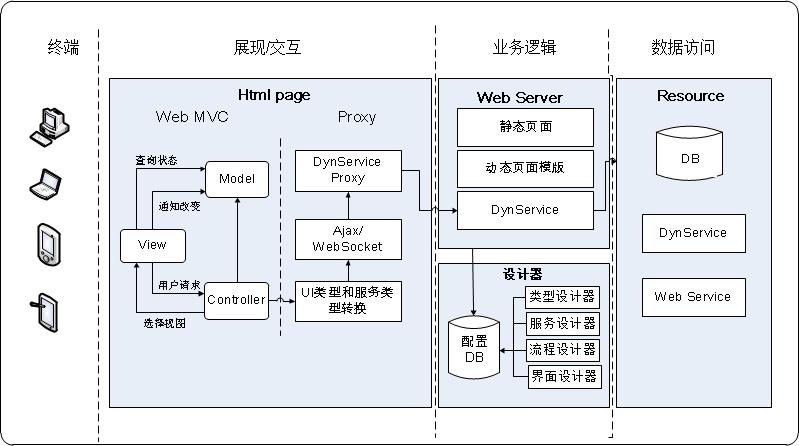
分布式架构和统一通讯

复杂用户界面的表示和呈现

完成了上述的几个要素后，该平台即可用于一般的基于数据库的业务系统开发，如进销存、资产管理等各类信息系统的开发，也可用于在线生产监控系统的开发，也可用于IT监控和自动化系统的开发。

开发一种基于动态对象和服务的快速开发平台，能动态建立一个业务系统所需要创建的领域模型和服务体系, 打造一个高效的、可长期持续发展的业务开发基础设施，基于该框架，减少重复开发工作量、缩短开发时间、降低开发成本，可快速实现业务系统的应用逻辑。

## 平台基本架构：



系统建模采用领域建模的开发模式。基于模型驱动的系统，业务人员不需要编写代码，仅仅依据需求对业务实体、业务规则、工作流程、用户界面、外部服务进行建模，并将模型整合后，系统会在运行期间，依据模型动态形成业务系统。其开发、变更都非常容易，并且很容易实现界面跨平台，跨语言的能力。

平台运行时提供了面向服务的架构，面向服务的架构体系简洁高效，容易扩展，易于支持多种客户端，方便系统间做功能集成。

面向服务的[体系结构](http://baike.baidu.com/view/1188494.htm)（service-oriented architecture，SOA）是一个组件模型，它将应用程序的不同功能单元（称为服务）通过这些服务之间定义良好的接口和契约联系起来。接口是采用中立的方式进行定义的，它应该独立于实现服务的硬件平台、操作系统和编程语言。这使得构建在各种这样的系统中的服务可以以一种统一和通用的方式进行交互。对 SOA 的需要来源于需要使业务 IT 系统变得更加灵活，以适应业务中的改变。通过允许强定义的关系和依然灵活的特定实现，IT 系统既可以利用现有系统的功能，又可以准备在以后做一些改变来满足它们之间交互的需要。

系统的服务层内核是基于领域建模的OO体系，提供了稳定可靠的业务内核模型，对外包装为SOA体系，带来了很多灵活性，这种结合带来巨大的好处。

## 客户端技术

快速开发平台需要提供富Web客户端(RIA)前端界面，注重UI与互动，基于mvc模式开发。前端技术发展迅速，并且比较复杂，为避免重复造轮子，前端开发可关注已有的开源前端mvc框架，前端mvc框架应与组态模型驱动开发理念结合在一起。

为提高客户体验，尽量强化客户端技术，包括完全客户端技术、混合客户端技术。完全客户端技术就是彻底强化客户端，页面只是个静态的html，通过引入大量的js实现客户端的布局、数据填充、交互和验证，服务端基本作为数据源使用，通过ajax技术，可以实现非常类似cs体验的web应用。混合客户端服务端技术，用户布局界面（也可以加上初始的数据填充）是服务端生成的，但是数据动态填充、交互、验证更多的使用ajax技术。

由于客户端交互比较复杂，系统开发过程想实现快速和灵活相结合的目标，从而需要采用支持动态生成,基于模型驱动的组态技术，该技术没有显式的编码阶段。通过一组设计器工具生成通用界面描述，运行期利用动态模版解析，动态形成界面。基于模型的开发模式使前端应用是动态生成的，可通过不同的视图模型（即换用不同的模版）适配不同的计算平台，如pc、平板、手机等。后端则可直接跨平台共用。

对象模型

设计器

模版

解释器

页面

图1、模版驱动自动界面技术

设计器在设计实体对象模型时,会考虑已经抽象出来的表单页面模型的类型, 不同的表单页面模型会对应通的界面设计信息的标记,后台模板解析引擎会根据这些标记和表单页面模型的类型生成对应的前台页面代码。

## 服务端技术

后端主要提供数据访问能力，注重接口定义清晰，业务逻辑实现简单，数据访问安全高效，可以通过多种缓存技术提供性能。

服务端技术主要是安全验证和数据通讯。客户端ajax需要将服务端视为数据源，如果没有很好的安全验证，恶意调用服务端服务可能造成数据泄密，恶意删除修改数据的可能，因此需要找到很好的办法防止不安全的服务提供。通常服务端通过提供WebService等方式提供数据，为了支持组态方式的快速开发模式，服务端应该支持动态数据源。也就是说服务是可能随时组态进去的，随时可能变更服务，没有静态发布的过程。

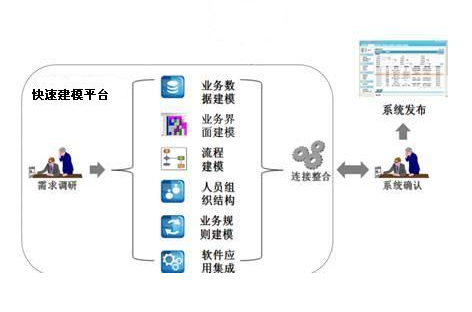
后端的开发技术可在java、.net中选择，其中java和.net都属于成熟开发语言，有很完善的类库和开发环境支持，根据开发者的习惯任选一种即可。

服务端包含一组引擎将设计器设计的XML模型+一些模版动态形成服务或者用户界面。由于服务端需要链接其他的资源，如其他系统的API，外部Webservice等，因此需要提供一些Adapter。考虑到特殊场景需要访问异构网络的远程RPC资源，服务端支持对服务的集中管理，通讯应支持应用层面的路由和转发功能，通讯内部使用消息队列方式。

为确保信息安全，服务端需要对客户端请求进行身份验证，确保权限合适的用户才能访问服务。用户登录时，向Session服务器申请唯一的会话ID（SessionID），在后续的页面中，SessionID会添加到请求的头部，服务端截取SessionID后验证权限，有权可以执行调用。计划采用专用Session服务器的方法，可以跟踪客户的登录、页面的访问，并记录日志。

## 设计器及模型表示

基于模型驱动的系统，业务人员不需要编写代码，仅仅依据需求对业务实体、业务规则、工作流程、用户界面、外部服务进行建模，并将模型整合后，系统会在运行期间，依据模型动态形成业务系统。其开发、变更都非常容易，并且很容易实现界面跨平台，跨语言的能力。



基于模型驱动的快速开发模式

快速开发平台分为两级模式：开发人员视图模式和最终用户视图模式。开发人员视图模式包括动态类型，动态服务等基本对象。系统先提供开发人员模式，尽管该模式中的概念一般用户不容易理解，但是开发人员能够理解，并能实现快速开发的目的，最终用户模式使用更简单的概念，使用向导等直观方式完成主要属性的设定，其他属性采用默认值，但是提供的功能选项相对较少。

提供可视化的UI设计工具及对象模型设计工具。在模型和静态模版基础上可由机器自动生成页面可执行代码，允许人工修改生成的页面并注册到系统中去。

主要的设计器有如下几种：

业务对象设计器：设计诸如客户、供应商这类的业务对象，包括定义对象的属性方法等，对象之间会出现关联关系，如订单和订单明细具有聚合和组合关系，这类关系用来保障用户界面向下和向下的关联导航及业务一致性约束。对象之间的继承关系用来处理分类系统，如资产管理的各种类别的资产的管理。部分需要存盘的业务对象动态配置后，可以动态形成和变更数据库中的表，实现无编码方式的数据库建模和业务对象建模。

界面设计器：每种表单页面模型会有一组界面对应，如DictForm字典页面模型，BillListForm单据列表页面模型, BillForm单据的维护页面模型, TreeForm树形结构的对象页面模型。通过拖拽布局形成界面模型，由模板解析引擎根据模型和模板进行动态解析生成可执行的页面代码.

流程设计器：用来将孤立的业务处理作为环境连接为一个完成的流程。每个业务环节会对应一个执行者组和表单界面，可视化设计器能支持流程的动态变更和版本控制。

服务设计器：服务设计器用来设计动态的数据服务，形成服务契约(服务接口)。服务分为两类:

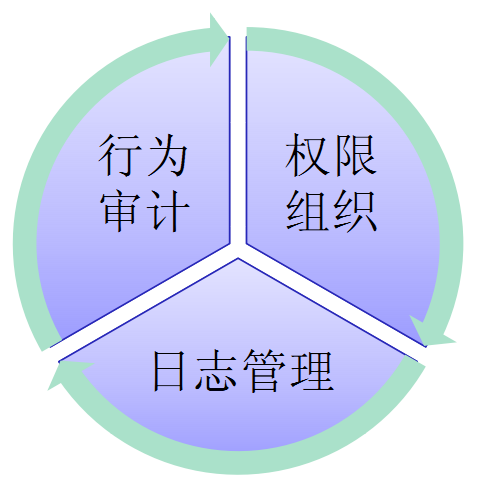
一类是通用的服务,这部分完成系统中抽取出来的和业务无关的一些通用服务,例如一些基本对象的增删改查,这些基本的增删改查不需要特定的业务逻辑代码,也不需要进行对象间的关联修改,类似这样的代码都纳入到通用的服务中去,通用的服务在系统模块注册,系统初始化时进行动态加载。

另一类是和系统的业务息息相关的服务,包含对象本身的增删改查,及业务所需要的关联关系的增删改查,这类服务的具体实现,千差万别,不可能抽象出通用的代码,所以在系统设计时设计出统一的服务接口,由开发人员根据接口实现具体的业务逻辑,这些实现的业务逻辑代码都封装成动态链接库,在系统初始化时通过反射加载,供系统调用。

## 组织建模与权限控制

一个用户登录不同系统就需要采用多个账号、密码，使用不便，各个用户登录不同系统就需要采用多个账号、密码，使用不便。同一用户使用信息系统中，如果涉及多个子系统，需要重复的进行登录。不同系统都采用自身的一套认证和用户管理机制，不利于统一管理。

有些用户离职以后由于缺乏统一管理，仍然能登录一些业务系统。面对很多样式的权限管理模型，管理员也无从下手。管理员需要管理十几甚至几十套应用系统的用户。



根据以上需求，设计统一权限控制模块。平台实现 RBAC（Role Based Access Control）的基本思想，在此基础上，为企业应用系统中提供 R-F-RBAC（Role-Function-Resource Based Access Control，基于角色-功能-资源的权限控制）的一种具体方法。

* 一个角色可定义多个功能或资源的操作能力。
* 一个用户可归属于多个角色，同时继承用户组的能力。

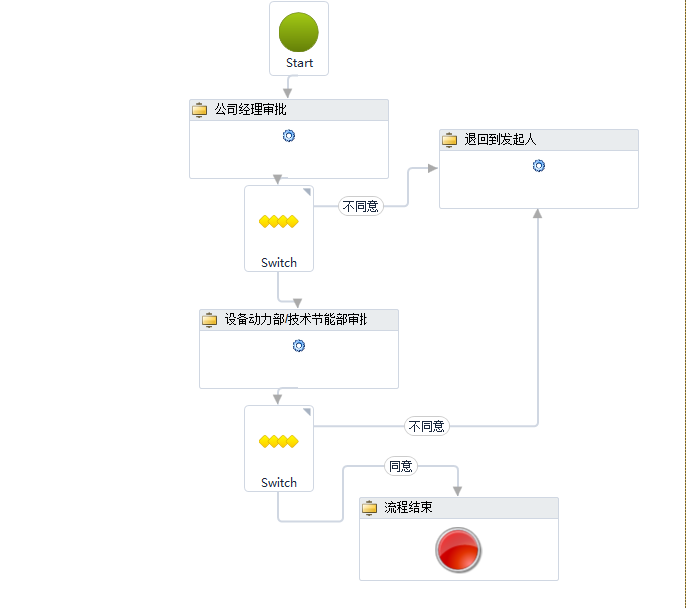
 功能或资源操作定义采用了：允许或拒绝的方式。如果对同一功能同时存在允许和拒绝定义（不管是用户还是用户组定义，是直接声明或继承声明），系统采用拒绝定义。

## 工作流系统

工作流就是“业务过程的部分或整体在计算机应用环境下的自动化”，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传递文档、信息或任务的过程自动进行，从而实现某个预期的业务目标，或者促进此目标的实现”。

工作流系统主要由流程定义模块、流程引擎、流程监控模块三个大的部分组成，其中流程引擎是核心。

平台的工作流系统对业务流程的设计、执行、监控、分析、改进提供全面支持。帮助用户在业务过程中，全面记录过程流、协作流、信息流的内容，帮助用户依据流程运行的事实和结果，进行有效的流程分析，以优化过程流的环节、降低协作流的时滞、压缩资源流的成本，并实现流程的改进。在技术上实现了工作流数据与业务流数据的统一，控制流与界面流的统一。也就是说，工作流系统不再是独立运行的系统，不再是需要大量与客户系统完成客户化的系统。



图示:工作流设计效果图

当工作流的流程边上变化时，相关的工作流界面将自动调整，也不再会发生流程调整很方便，但是调整了没有效果，因为承载它的业务系统还没有调整，所以流程调了等于没有调等情况。

平台提供工作流设计器为可视化的流程设计工具，用户通过拖放等方式来绘制流程，并通过对于环节的配置来实现环节操作、环节表单、环节参与者的配置。工作流设计器为用户以及开发商提供了快速绘制、修改流程的方式。

流程监控对流程执行过程进行监控，包括流程运转状况，每个环节所耗费的时间等等，而通过这些可相应的进行流程的优化，以提高工作效率。在监控功能还提供取消流程、重启流程等功能。

工作流表单用于承载流程数据的展现或是交互操作能力。平台直接使用实体表单提供工作流表单功能。利用其强大的表单设计能力，所见即所得的操作灵活性，与业务系统数据无缝连接为工作流系统提供了强大的数据展现及交互能力。

## 术语和定义

**领域模型概念**

* **实体**

以标识作为基本定义的对象称为“实体”。

在实体的生命周期中，其内容和存在形式都可能发生巨大的变化，但是必须确保的是标识的连续性，定义了实体的标识就可以有效地对其进行跟踪。

* **服务**

服务本身不属于实体，同样也不属于值对象；它只代表领域中的一些重要操作，是一些本质的行为和动作，除了它们掌握的操作外，服务没有自己的任何状态，在领域中也没有什么意义。但是，如果把这些领域需要的行为和动作交付给实体或者值对象的时候，就会破坏模型中对象的定义，人为添加了一些没有意义的对象。当领域中一个重要的进程或转换操作不是实体和值对象本身的职责时，把操作作为一个独立的接口加入模型，并声明为服务。

* **实体之间的关系**

实体关系类似于数据库领域的ER关系,表达了领域实体之间的常用关系,包括一对多,多对一两种,平台还支持继承关系。

**界面模型概念**

* **表单**

在业务处理过程中主要负责数据采集功能,向用户提供直观的数据展示,把采集到的数据提交到服务端进行处理。

由于不同领域的业务系统处理方式存在巨大的差异,所以表单的表现形式也多种多样,快速开发平台界面设计的核心就是综合各种系统的业务处理方式,对页面表单进行抽象,提供一组抽象的表单模型,平台在进行对象设计时充分考虑后台数据的业务处理,及前台的页面模型的类型,把前后台的信息融合到实体设计的过程中,这样后台根据这些信息,会自动的生成对应的数据库及对应的持久化信息,前台会解析这些信息生成对应的前台页面。

目前平台支持以下几种抽象出的表单类型:字典页面模型,单据列表页面模型,单据的维护页面模型,树形结构的对象页面模型.根据实际的业务情况平台还可以支持更多的表单类型,提供了良好的扩展性。

* **字典页面模型:** DictForm
* **单据页面模型:** BillForm
* **单据列表页面模型:** BillListForm
* **树形结构的页面模型:** TreeForm
* **多对多关联页面模型:** RelationForm
* **表单项:** FormItem

各类表单基本的编辑项.

* **查询项:** QueryItem

各类表单基本的查询项

* **表格:** DataGrid

各类表单中列表数据的容器

* **表格列:** GridColumn

表格中列的信息

## 参考资料

软件体系结构-模型驱动体系结构(MDA) ---北京大学软件工程国家工程研究中心

## 功能划分

* 对象设计
* 服务设计
* 流程设计
* 界面设计

## 常规界面规范

通过对各种业务系统中大量的界面布局和业务处理进行分析和研究,我们抽象出了几种最常用的页面模型,页面模型配合通过平台设计的实体共同工作实现了界面设计的核心技术-界面模板,通过界面解析器对模板进行解析,就可以自动生成页面所需的代码。

界面模板具有跨语言的特点,目前主流的开发语言的代码(C#,java,javascript……)都支持,界面模型的思路提供了良好的可扩展性,既可以像手动开发项目那样关注页面的细节,又可以像成熟的产品那样实现快速高效的开发实施。

* **字典页面模型:** DictForm
* **单据页面模型:** BillForm
* **单据列表页面模型:** BillListForm
* **树形结构的页面模型:** TreeForm
* **多对多关联页面模型:** RelationForm

## 设计规范

* 因为对象及对象间的关系通过设计器进行设计,所以对象的一些属性需要进行一定的约定,每个对象的标识属性都是对象类型名称+ID,属性的数据类型为Int32,如果对象有名称属性,也约定为对象类型名称+Name,属性的数据类型为string,这样平台在引用对象或者存在对象间的关系处理是就可以按照这个规范进行处理。

# 应用程序体系设计

## 简述

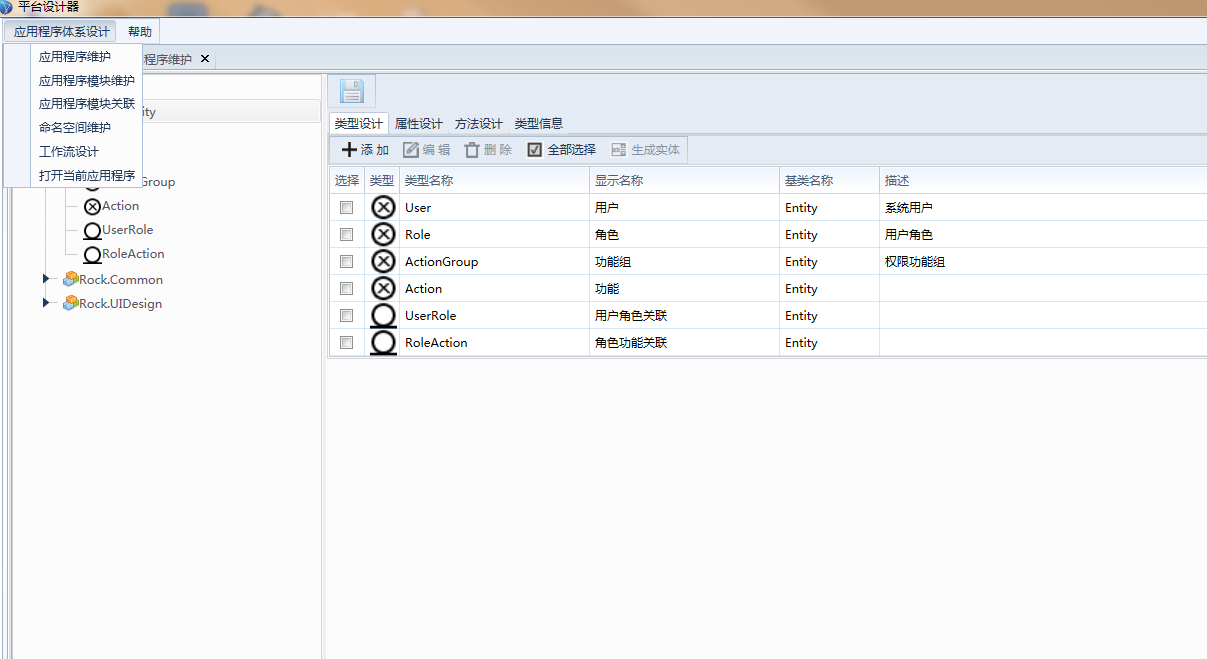
目前平台设计器分为两大部分,一部分是系统的体系设计,主要包括,对象类型及关联关系的设计,服务契约(接口)的设计,工作流的设计,这部分采用的技术是WPF+prism框架技术,实现了平台框架和具体的设计业务模块分离,框架通过容器注入具体的设计模块,各个设计模块之间充分解耦并提供了良好的扩展性。另一部分是界面设计器,目前该部分使用web页面进行设计,将来也会把这部分功能迁移到WPF+prism框架技术下,实现平台技术上的统一。

## 设计器主界面

所有的应用程序体系设计内容的设计都在这个页面框架下完成,

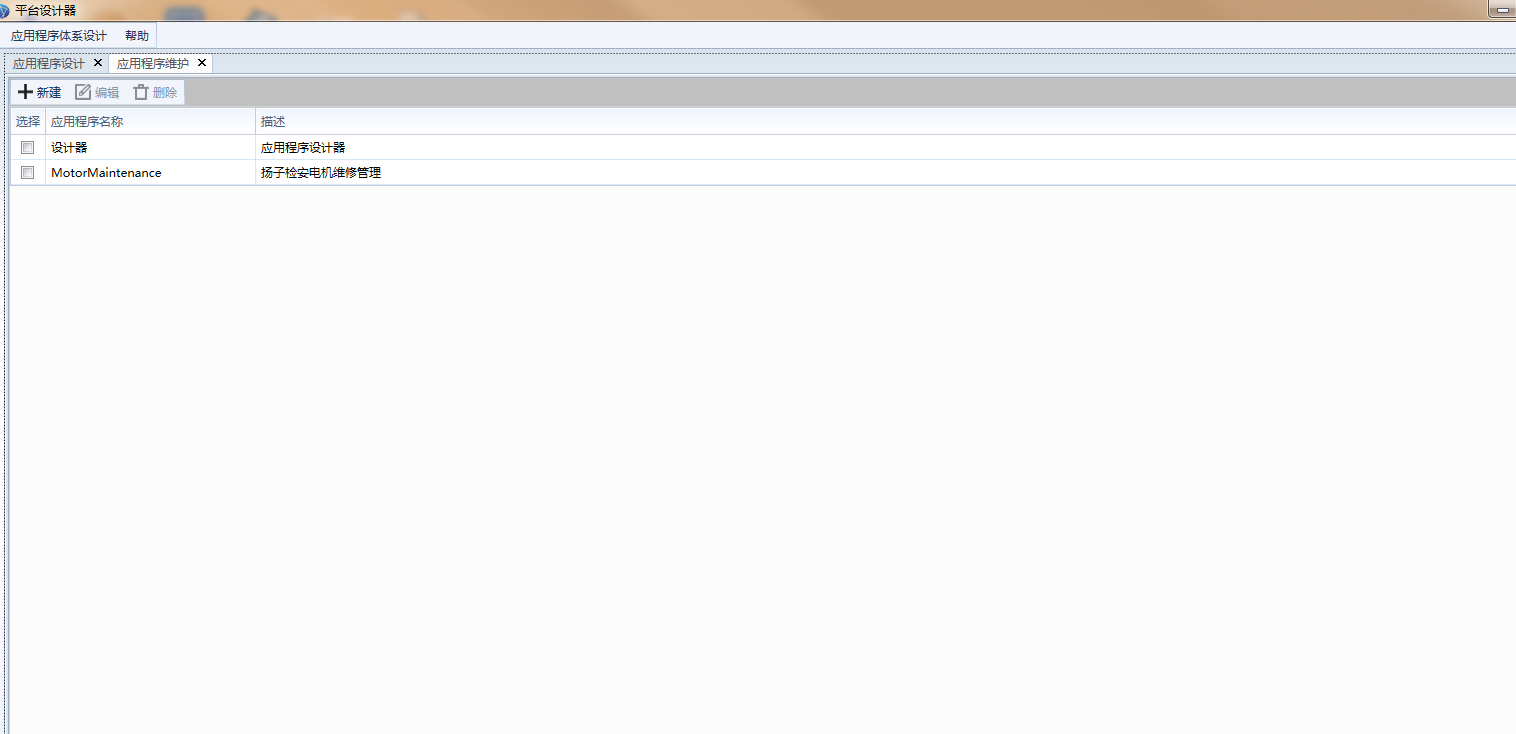
系统的总体结构:

* 应用程序:顶层是应用程序,开发一个新项目时,需要新建一个应用程序
* 应用程序模块:每个应用程序都由多个模块构成,模块和应用程序之间是多对多的关系
* 命名空间:每个应用模块下的所设计的实体类型的一个逻辑分类,类似于C#的命名空间
* 实体类型:应用模块下的业务实体类型,分为三类,实体类,关联类,控制类,实体类事由领域实体抽象出来的具体的类型,可以添加属性和方法,关联类主要是解决各实体之间的关联关系的类,控制类,是业务服务的契约,系统加载是映射成对应的接口,必须要有具体的实现,在系统初始化时反射加载,供系统调用。
* 工作流:平台对系统中复杂的业务处理提供工作流支持

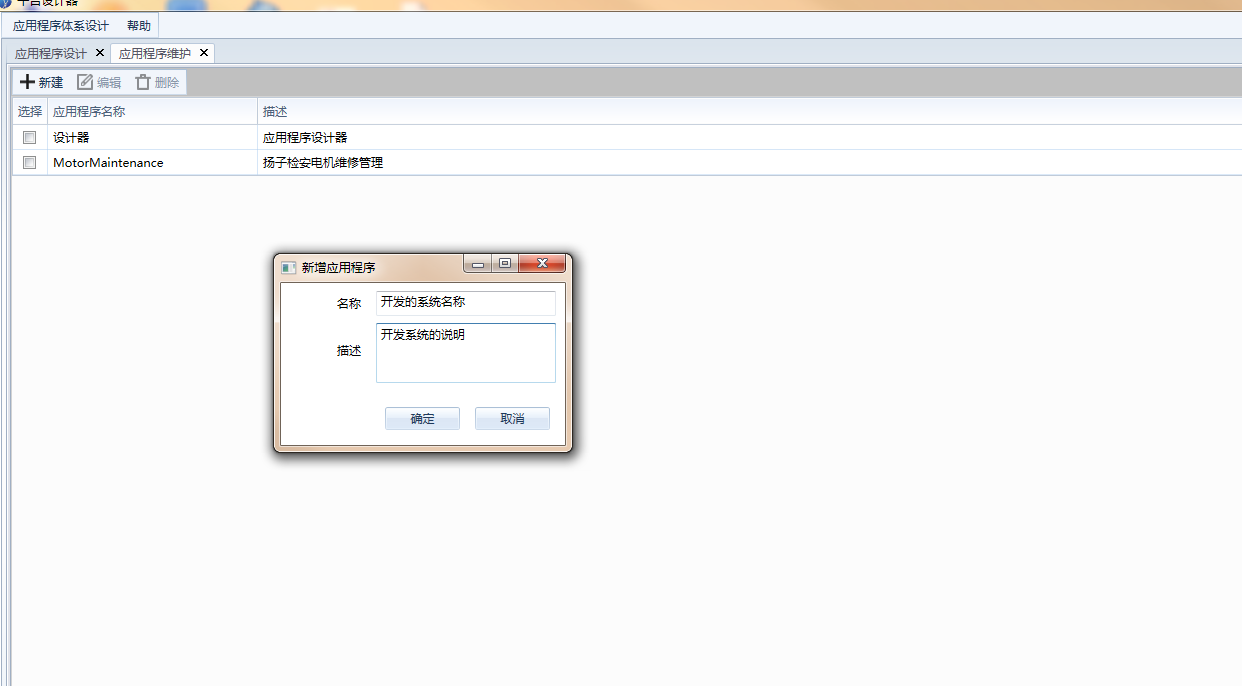


## 应用程维护

应用程序维护主页面

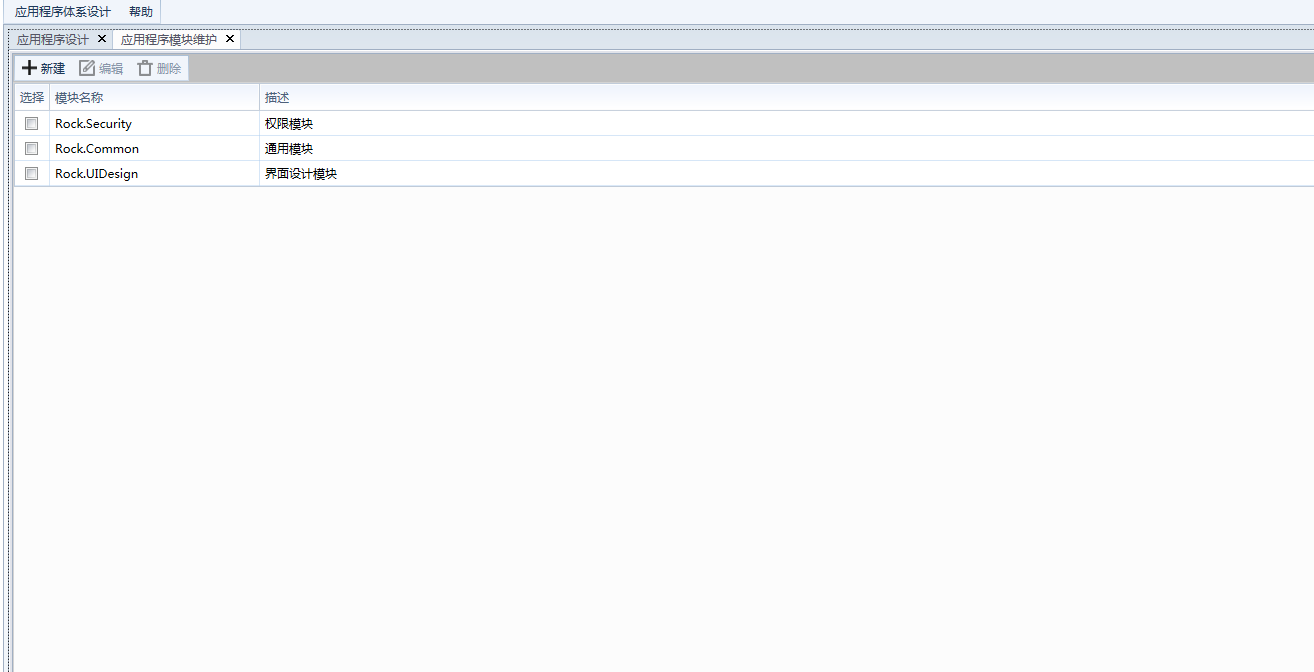


新建应用程序页面,选择具体的一个应用程序可以点击编辑和删除,实现编辑和删除对应的应用程序



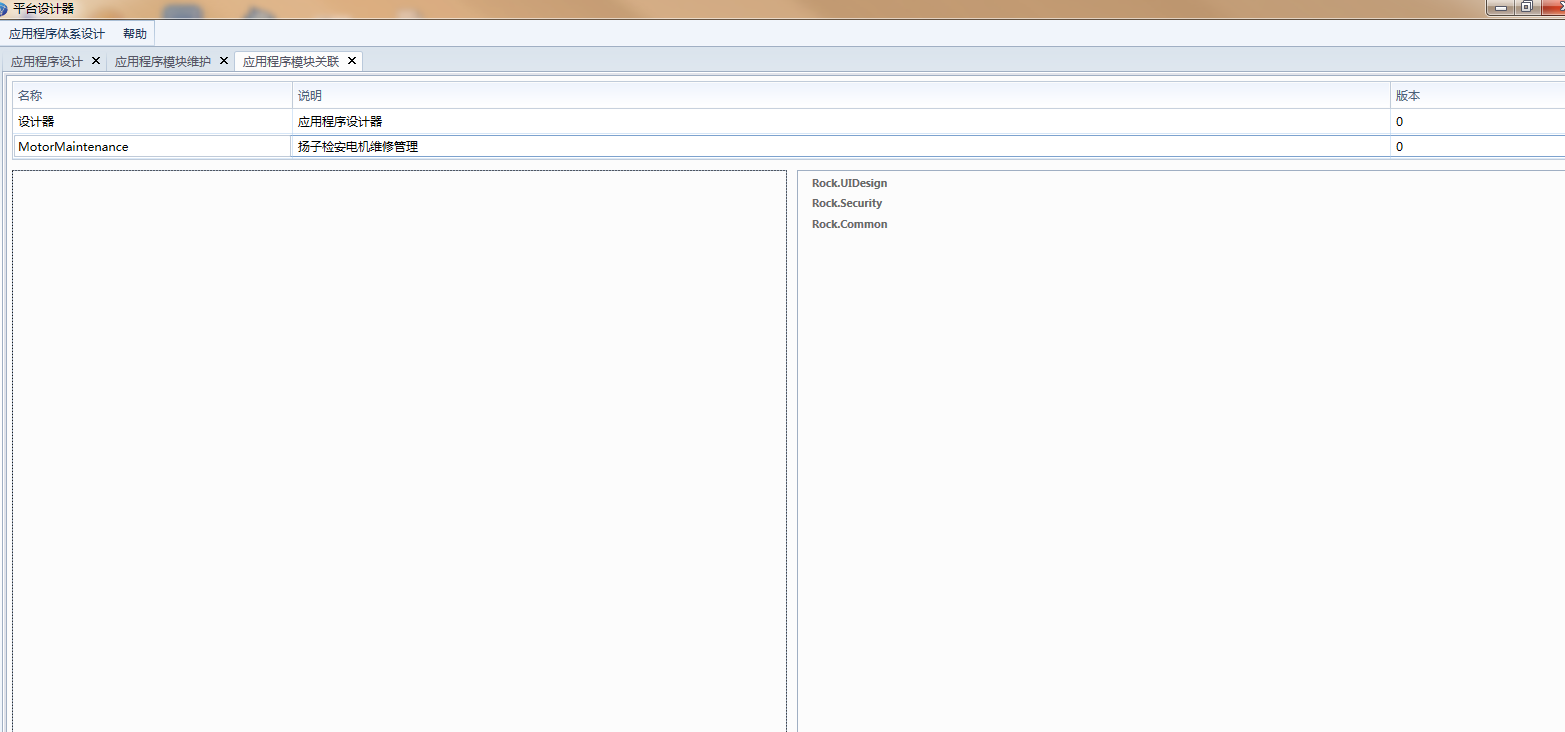
## 应用程序模块维护

应用程序模块是应用程序的组成部分,一个应用程序模块可以同时属于多个应用程序,其中权限模块和通用模块是每个系统都必须要选择的模块。



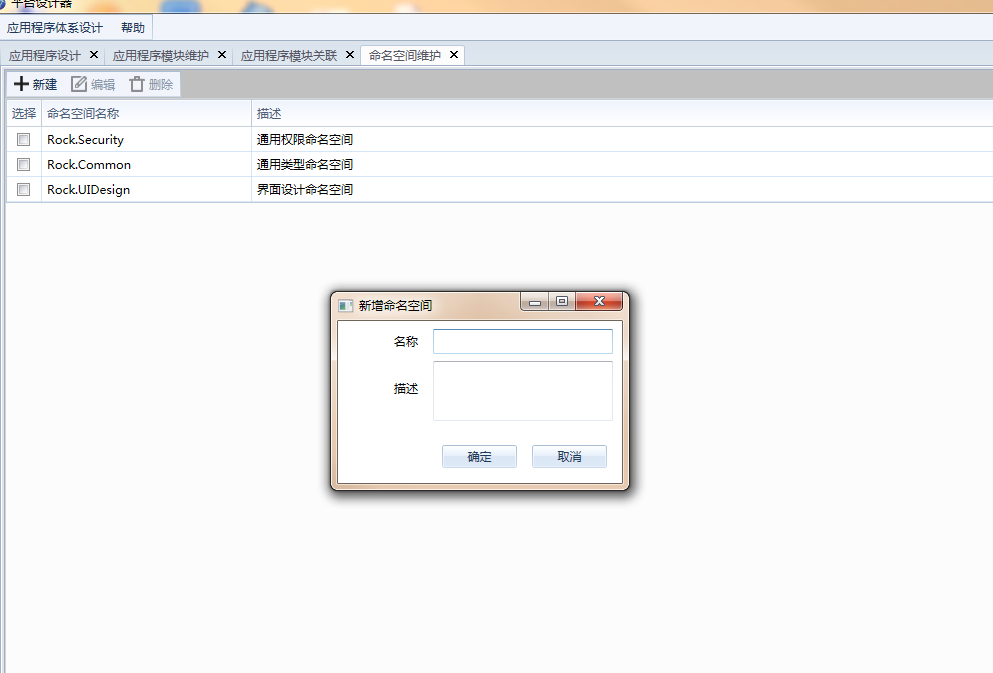
## 应用程序模块关联

该页面实现应用程序和模块的关联功能,左边的列表是未选的模块,右边列表是已选模块,可以用鼠标拖拽实现模块和应用程序的关联和取消关联。



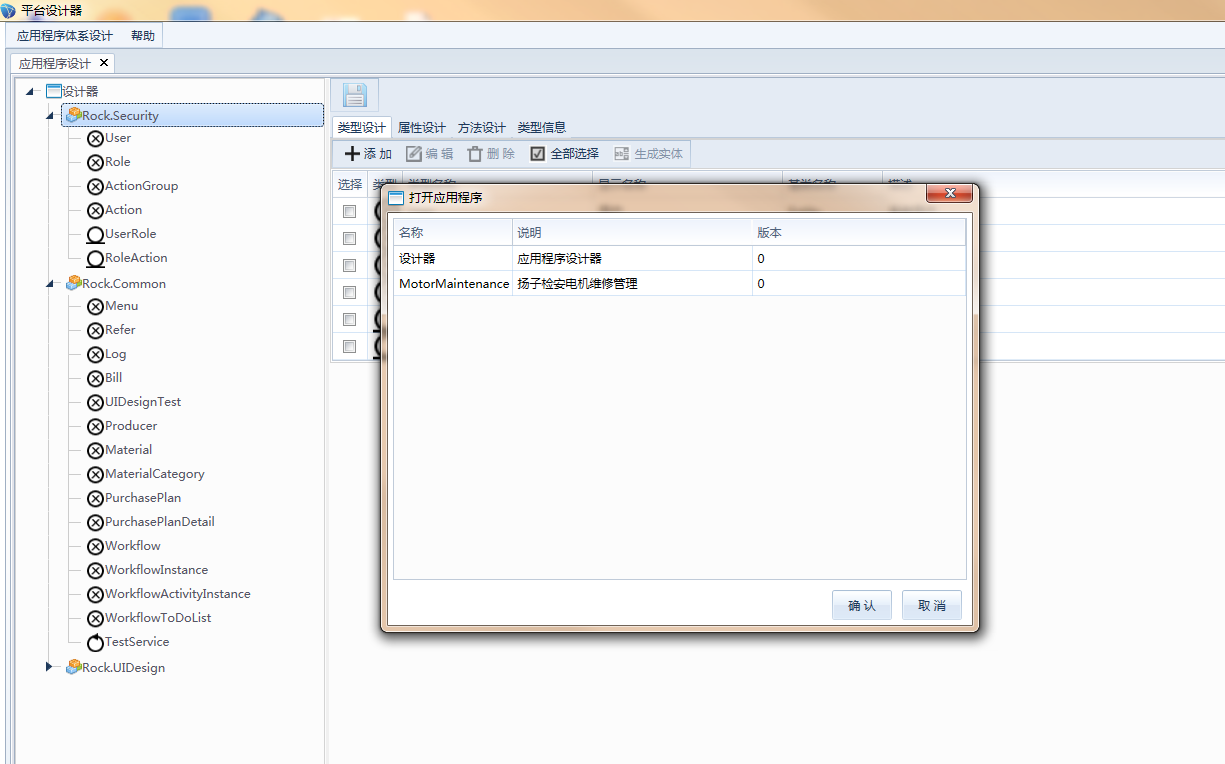
## 命名空间维护

命名空间维护页面,可以实现系统中命名空间的增删改查维护,已经被使用的命名空间不可以删除。



## 打开当前应用程序

在应用程序体系设计前需要先打开需要设计的应用程序作为当前应用程序通过选择然后选择确定按钮,或者双击所选应用程序都可以打开,应用程序打开后所有的设计操作都是针对当前应用程序的,默认打开的应用程序是应用程序设计器,这是一个最基本的应用程序仅包含权限模块和通用模块,没有其他的业务模块。

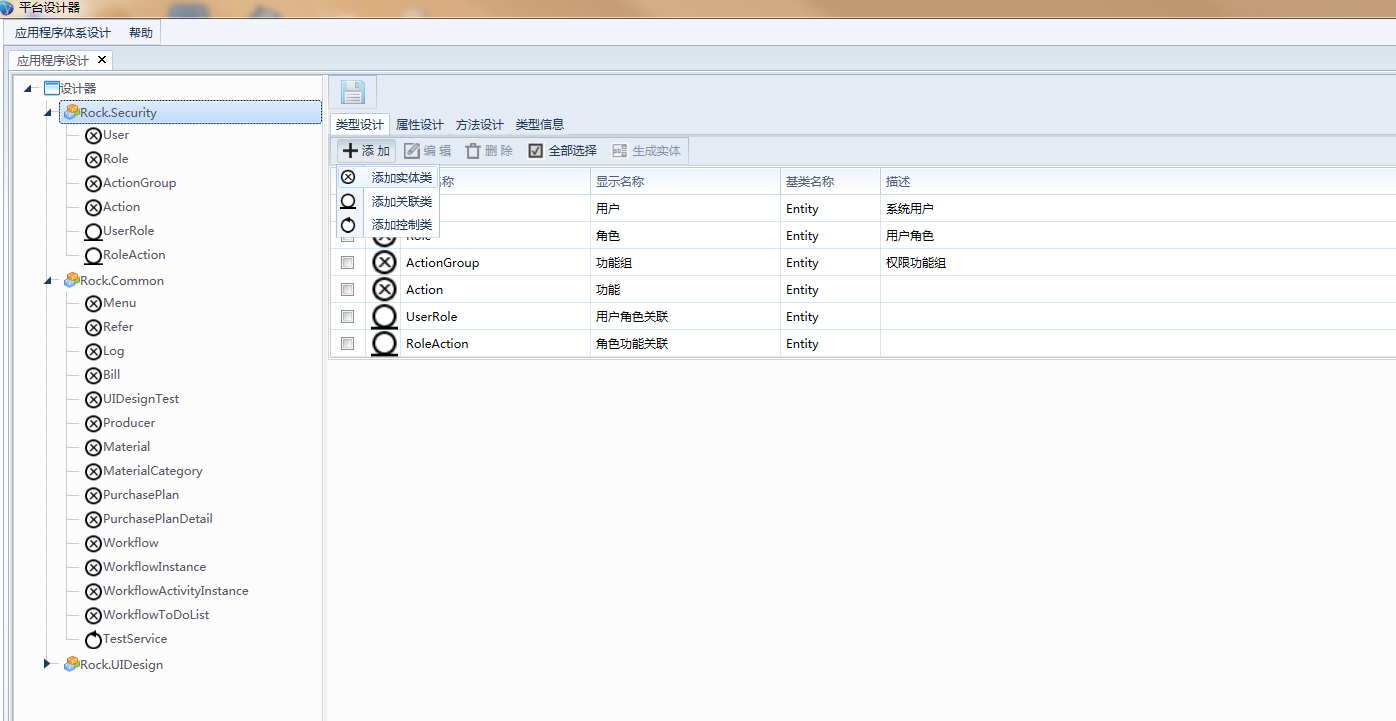


## 类型维护

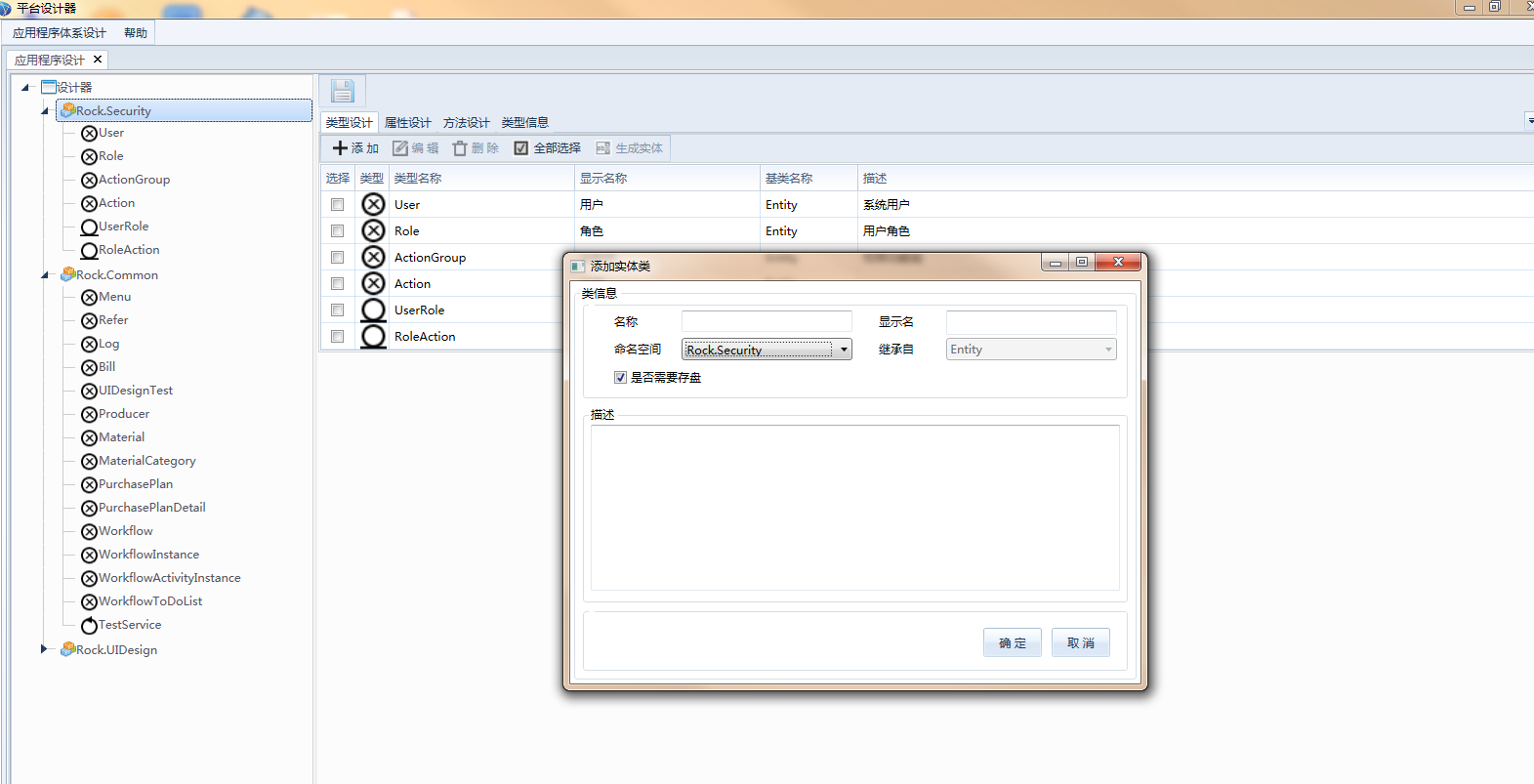
系统中的类型共分为三种类型:实体类,关联类和控制类, 实体类事由领域实体抽象出来的具体的类型,可以添加属性和方法,关联类主要是解决各实体之间的关联关系的类,控制类,是业务服务的契约,系统加载是映射成对应的接口,必须要有具体的实现,在系统初始化时反射加载,供系统调用。

### 实体类型维护

打开当前的应用程序后选择左边需要设计的模块,所设计的实体类型会添加到当前所选的模块中 ,点击添加按钮,会弹出一个下拉选择菜单,选择添加实体类进行实体类型的添加。

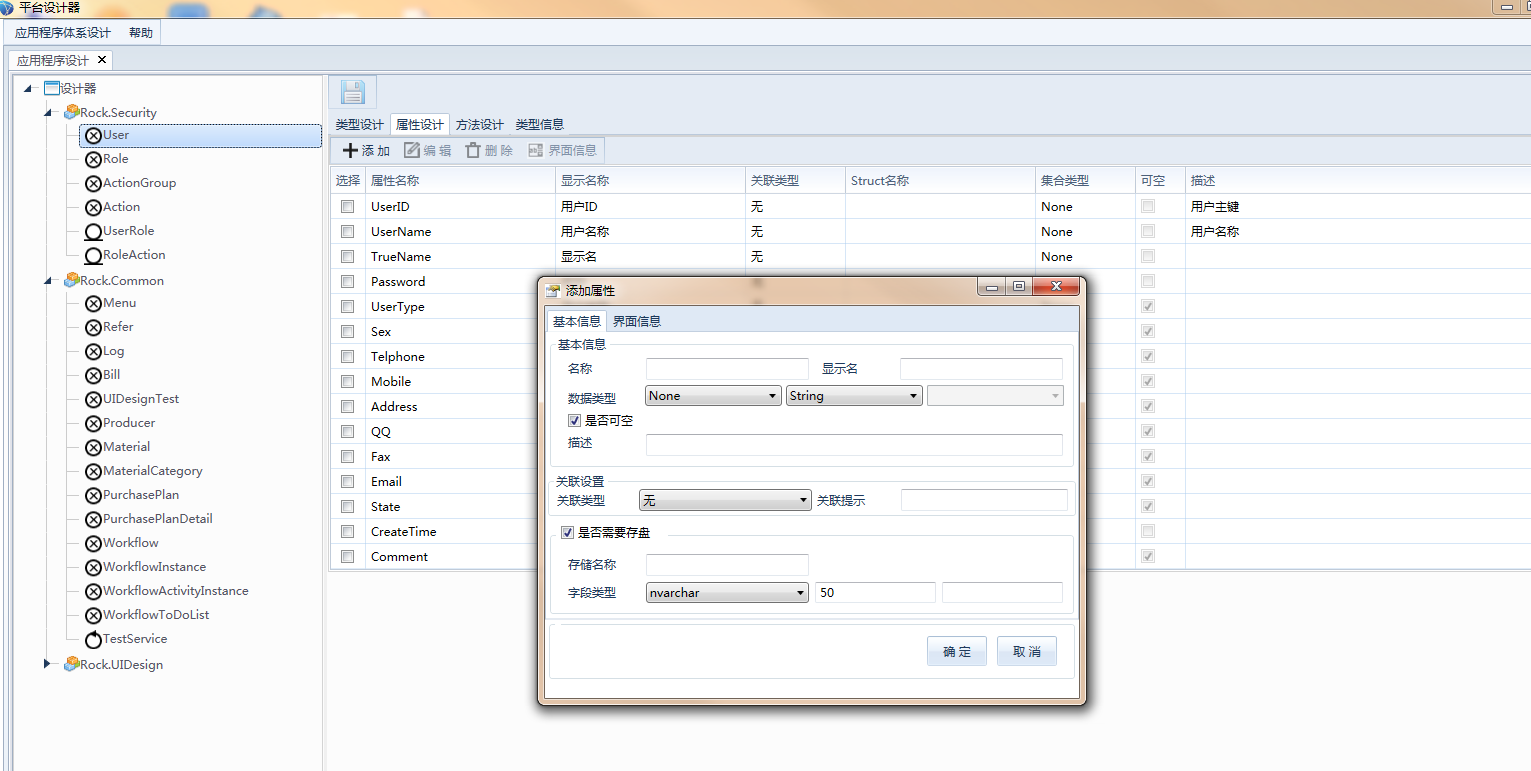


在弹窗中输入实体类型的基本信息点击确定按钮进入属性的和方法的设计过程。



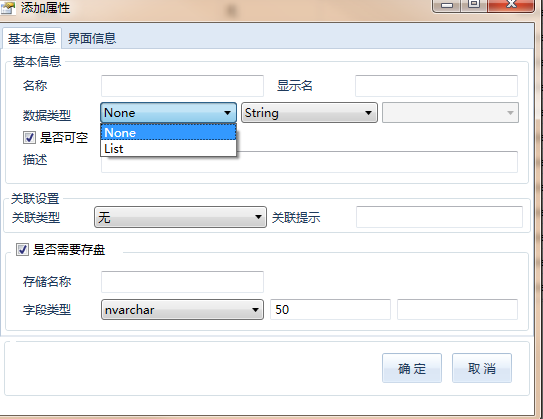
新增属性界面,属性设计界面包含两部分选项卡,基本信息,是对象自身的设计信息,界面信息是为界面设计模板解析引擎提供的界面设计信息。

基本信息说明:名称是属性的名称必选项,显示名,是后续界面设计时在用户页面上显示的中文名称,是否可空表示该属性是否可空,包括后台生成的数据库表的属性.是否需要存盘表示该属性是不是在后台生成具体的数据字段,如果不需要存盘数据库表中不会生成对应的字段,该属性仅在内存中存在,作为数据传输的载体。

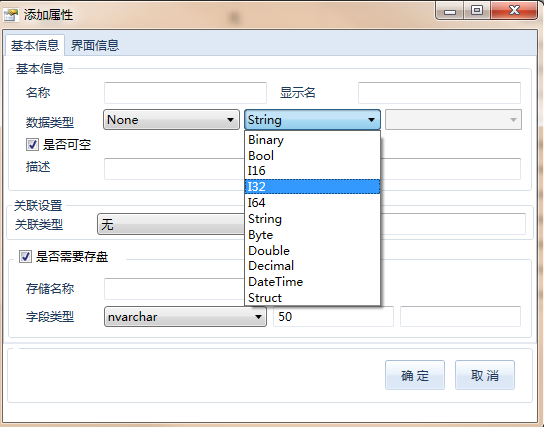


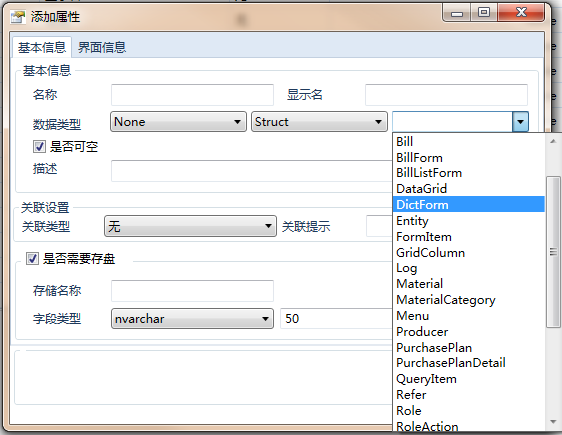
数据类型:

分成三个选择框第一个是指的属性的数据类型的集合属性,None是非集合属性。



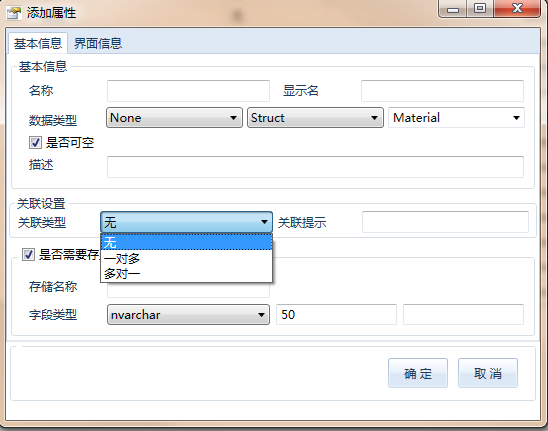
第二个框是指的属性的数据类型其中Struct代表该属性不是基本类型是对象类型,该列表中除了Struct以外都是基本数据类型,如果选择了Struct,就需要从第三个选择框中选择具体的对象类型类型。





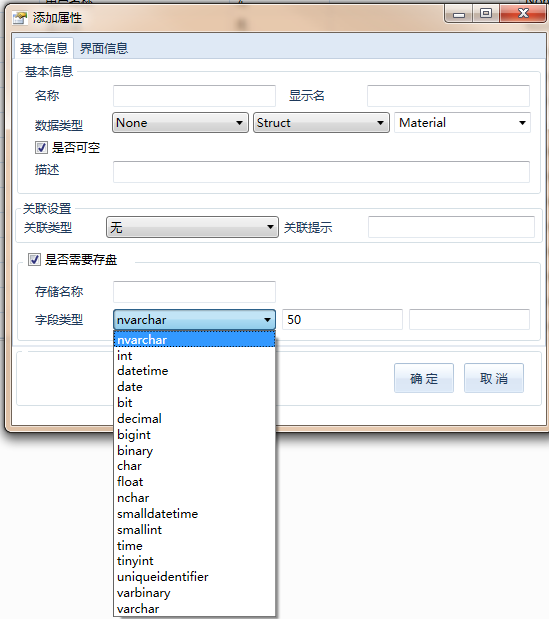
关联类型设置:如果当前属性是一个对象类型的属性,一般都会存在一个关联类型,一对多表示该属性是应用的一个具体的对象,类似数据库中ER关系的中的一对多,多对一在我们的设计中代表了是一个集合类型的对象关系,比如一个单据对象的明细集合列表,多对一的关系不可以进行持久化操作,不能存盘。

存储名称是生成数据库时表的字段名称,除了关联对象属性外这个属性一般直接取值名称属性不要进行更改,因为后台的业务处理及界面的模板解析引擎都默认采用属性的名称从数据库中加载对应字段的名称。



字段类型是指数据库的字段类型,其中decimal类型需要三个信息来表示,所以字段类型后面最后一个文本框是用来输入数据的小数点后保留位数的。

如果是字符串类型一般都有一个长度,默认长度为50,如果是文本类型的属性本系统采用Nvarchar+长度为0来处理,后台会制动处理成nvarchar(MAX)的数据类型。



界面信息:

界面信息是为前台页面设计器提供的后台设计信息,

**是否必需输入**:表示页面的表单项是不是必需输入的字段。

**是否只读**:如果只读,则根据属性信息生成表单页面时进行只读标记,用户不能手动输入信息,只能后台代码进行赋值。

**列表标题**:表示生成表单页面中的表格列的头信息。

**列表宽度**:表示生成表单页面中的表格列的宽度信息。

**列对齐方式**:表示生成表单页面中的表格列的内容的对齐方式。

**列排序式**:表示生成表单页面中的表格列的内容的排序依据。

**列类型**: 表示生成表单页面中的表格列的显示类型,ro表示只读,ch表示checkbox,ed表示可以在表格中进行编辑。

**验证类型**:表示生成的表单页面中的表单项提交时进行数据验证的数据类型,默认是None,目前支持数字和日期,如果设置成数据和日期,表单提交时系统会自动检查所提交的数据是否合法,目前仅支持数字和日期,后面根据需要可以随时扩展。

**参照类型**:如果该属性的取值来源是数据库中的通用参照表Refer则需要填写Refer中的对应ReferType信息以便模板解析引擎在解析时进行合理的处理,如果该属性的取值来源不是来自通用参照表Refer则该信息无需填写。

**页面控件类型**:生成表单页面的表单项的类型。

**查询方式**:属性的数据查询方式,包括关联对象的查询方式。(Fuzzy,Date,Value为自身数据的查询方式, Combox, Tree, Quick为关联对象的查询方式)开发人员在进行实际的开发选择是一定要遵循下面所列出的设计说明,如果不按照规则进行,会产生一些意想不到的错误结果。

**查询方式具体规则的设计说明:**

参照类型,查询方式,页面控件三个属性最终确定了表单页面的属性查询方式,查询分为表单项查询和查询项查询,他们的处理方式不同。(开发人员必需熟练掌握以下规律,才能进行快速正确的开发)

**表单项查询**:指的是在用户进行表单维护时所选择的对象属性信息,信息的来源分成三类

* 用户手动输入的信息,此时参照类型保持空值,无需考虑查询方式所选的值,这种情况平台后台对此不做判断条件,目前系统支持的可以搭配的页面控件有:Textbox(文本类型和数字类型)Data(日期类型)CheckBox(布尔型选择项)
* Refer通用参照表的信息

对象属性的值的来源是Refer通用参照表,此时参照类型必需填写Refer表中的ReferType内容,可以搭配的查询方式只有:Combox和Tree

* Combox表示数据源是非树状结构的通用参照表中表中的数据,通过表单页面的下拉列表进行选择,可选的页面控件只有Combox.
* Tree表示通用参照表中的数据是树状结构的数据,用户选择时会弹出一个弹窗,以树形结构的方式选择数据源,可选的页面控件只有Textbox.
* 独立的参照表的信息

参照类型必需保持空值,可以搭配的查询方式有三个Combox,Tree和Quick

* Combox表示数据源是非树状结构的独立数据 源,通过表单页面的下拉列表进行选择,可选的页面控件只有Combox.
* Tree表示数据源是数据是树状结构的独立数据,用户选择时会弹出一个弹窗,以树形结构的方式选择数据源,可选的页面控件只有Textbox.
* Quick表示数据源是数据是非树状结构的独立数据,用户选择时会弹出一个弹窗,用户输入快查关键字过滤所选择的数据源,可以实现数量较大的数据源的快速选择,可选的页面控件只有Textbox.

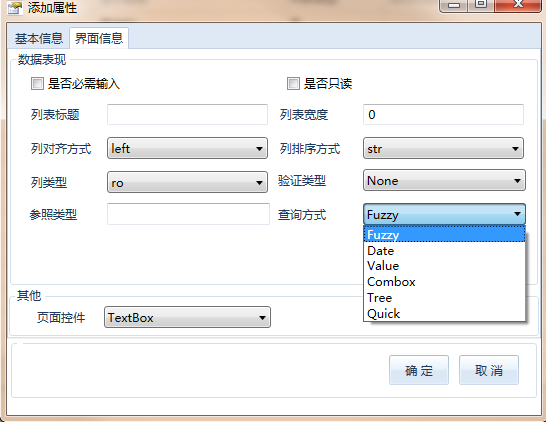
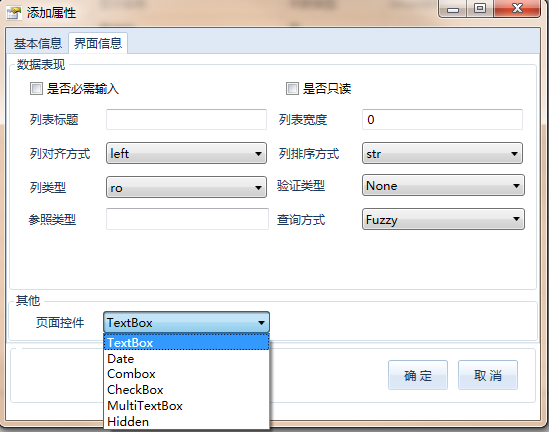
**查询项查询**:指的是表单页面进行数据查询时使用的查询条件,查询的条件的来源分成三类

* 对象自身的属性,查询条件的内容来自所在的表的对应字段,此时参照类型一定是空值,可以搭配的查询方式有Fuzzy,Date,Value
* Fuzzy表示查询的方式为模糊匹配的查询方式,对应的SQL语句为Like,可适用页面控件只有Textbox
* Date表示查询的方式为日期范围查询,对应的SQL语句为Between,可适用的的页面控件只有Date
* Value表示等值的查询,对应的SQL语句为 =,可适用页面控件只有Textbox
* Refer查询的条件的来源是从通用参照表选择到对象对应字段上的值,此时参照类型一定不可空,可以搭配的查询方式有: Combox, Tree
* Combox表示数据源是非树状结构的通用参照表中表中的数据,通过表单页面的下拉列表进行选择作为查询条件,可选的页面控件只有Combox.
* Tree表示通用参照表中的数据是树状结构的数据,用户选择时会弹出一个弹窗,以树形结构的方式选择数据源作为查询条件,可选的页面控件只有Textbox.
* 独立的参照表的信息

参照类型必需保持空值,可以搭配的查询方式有三个Combox,Tree和Quick

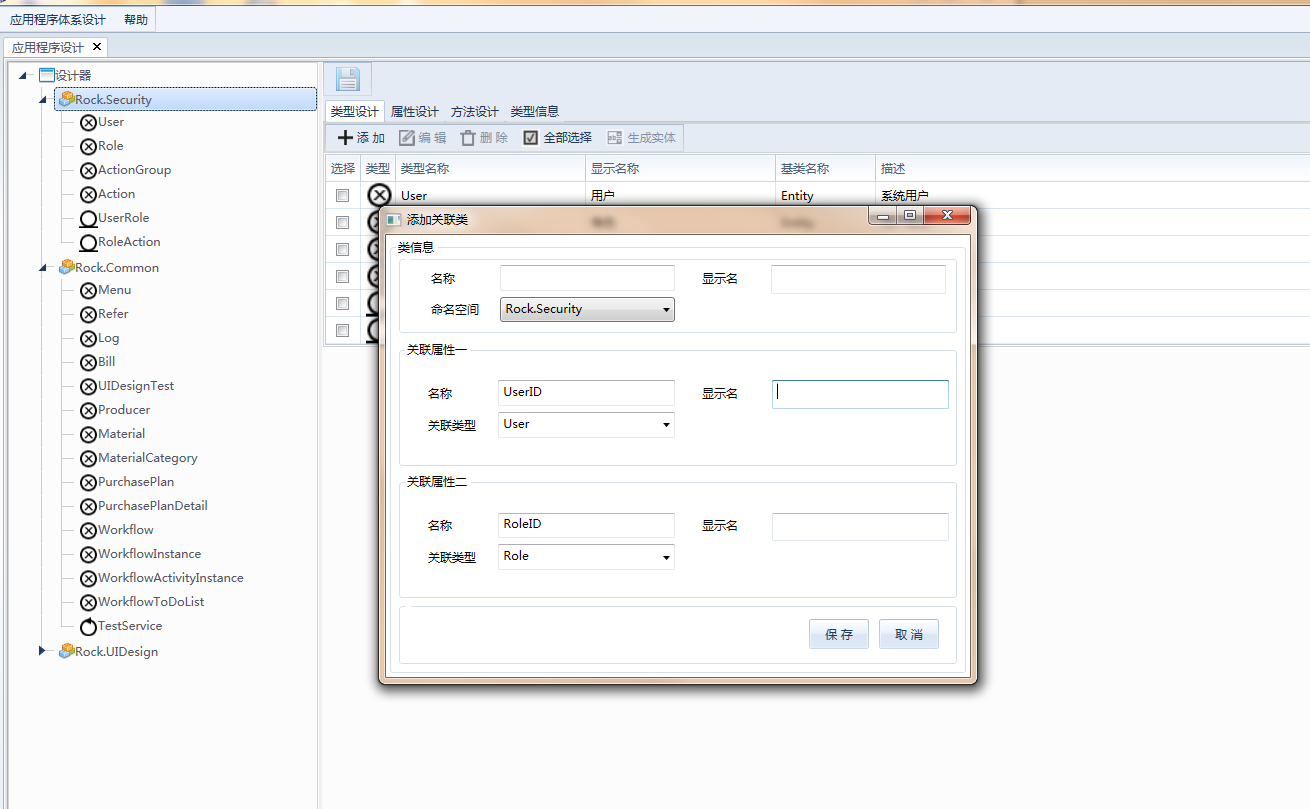
* Combox表示数据源是非树状结构的独立数据源,通过表单页面的下拉列表进行选择作为查询条件,可选的页面控件只有Combox.
* Tree表示数据源是数据是树状结构的独立数据,用户选择时会弹出一个弹窗,以树形结构的方式选择数据源作为查询条件,可选的页面控件只有Textbox.
* Quick表示数据源是数据是非树状结构的独立数据,用户选择时会弹出一个弹窗,用户输入快查关键字过滤所选择的数据源作为查询条件,可以实现数量较大的数据源的快速选择,可选的页面控件只有Textbox.



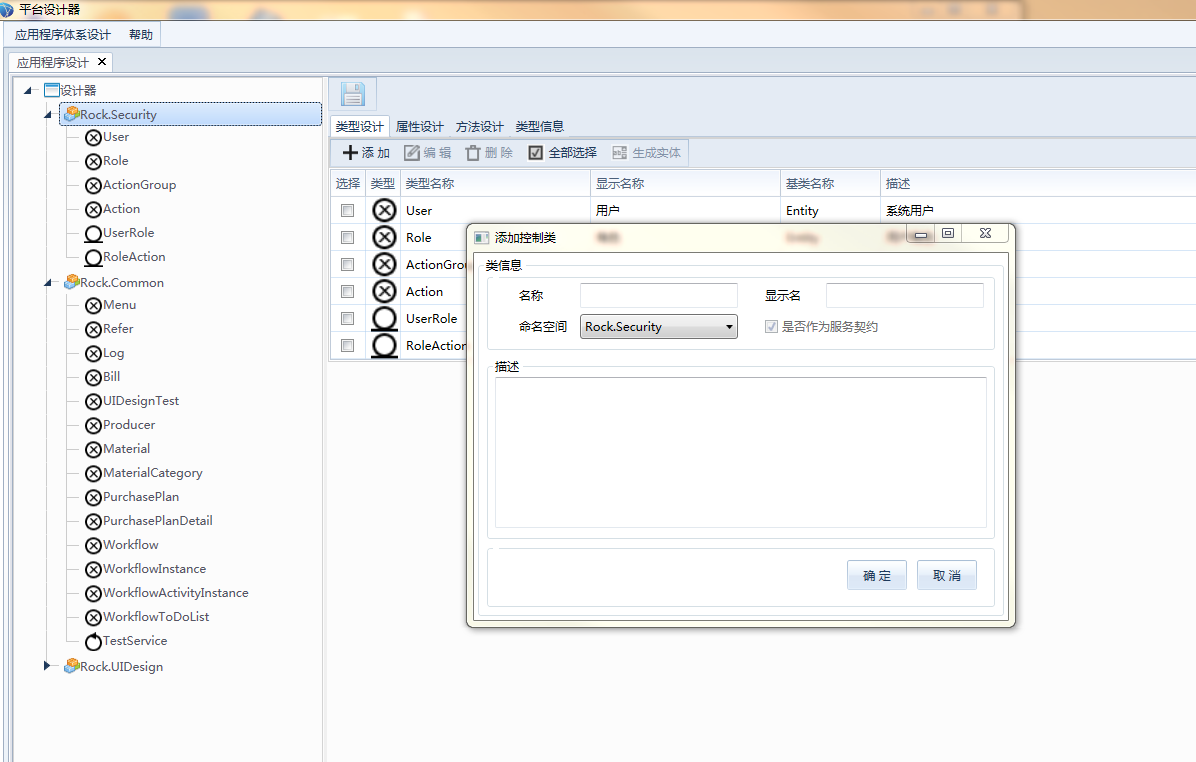
### 关联类型维护

关联类型是实体间多对多关系的一个抽象,通过关联类系统可以在自动生成关联关联关系的表结构,并且关联的对象之间自动添加约束关系,目前系统仅支持两个对象的关联关系.

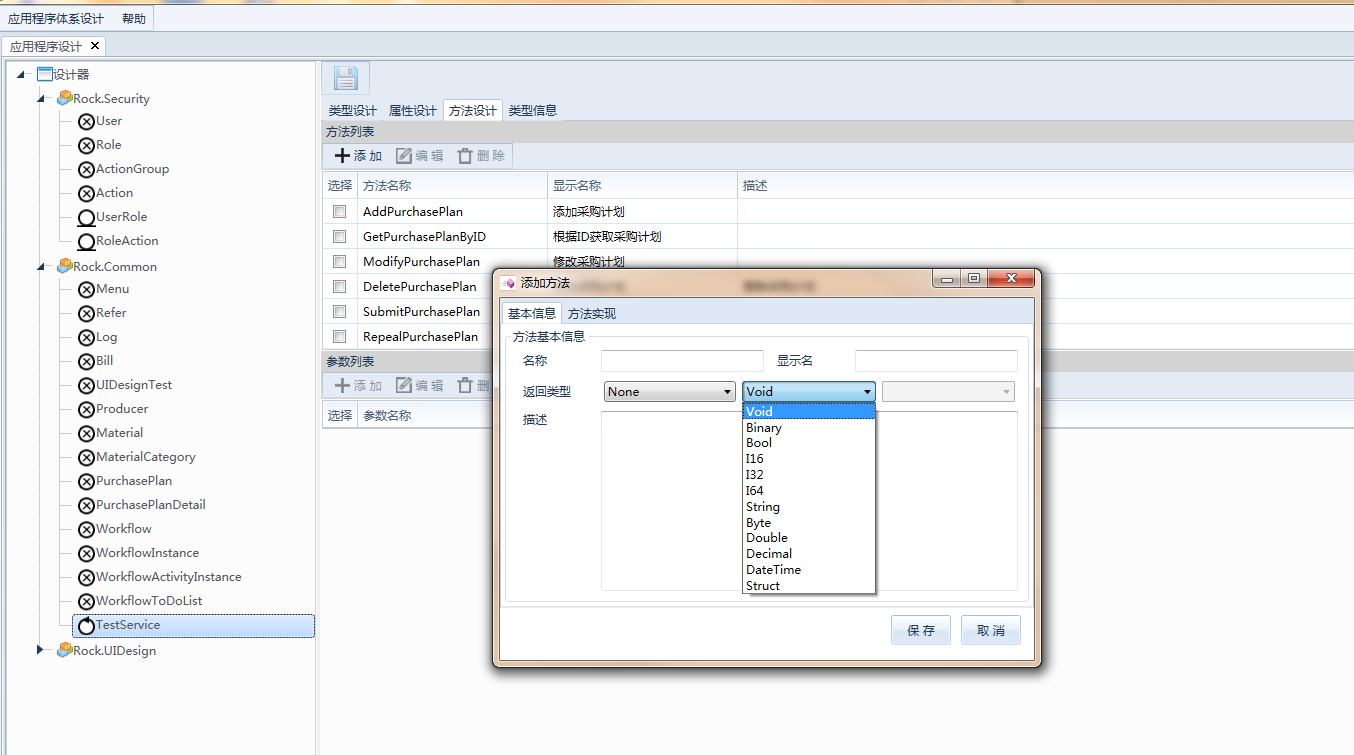


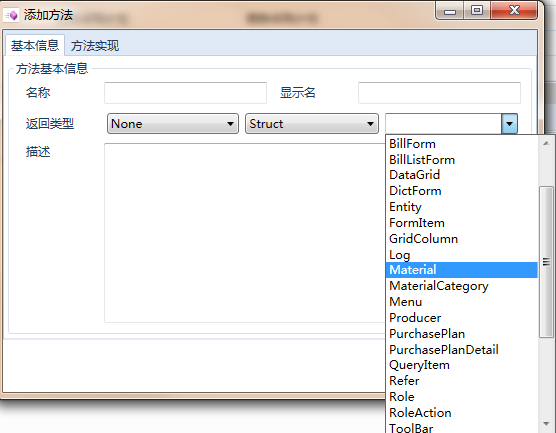
## 控制类维护

控制类是业务服务的契约,系统加载是映射成对应的接口,必须要有具体的实现,在系统初始化时反射加载,供系统调用。



控制类中接口的维护,返回类型的分成三个选择项,第一个是返回值的类型是否集合类型,如果是集合类型则选择List如果不是集合类型则选择None,第二个选择项是返回值的具体类型,除了Struct之外都是值类型,如果选择了Struct,就需要选择第三个选择项中的具体类型.





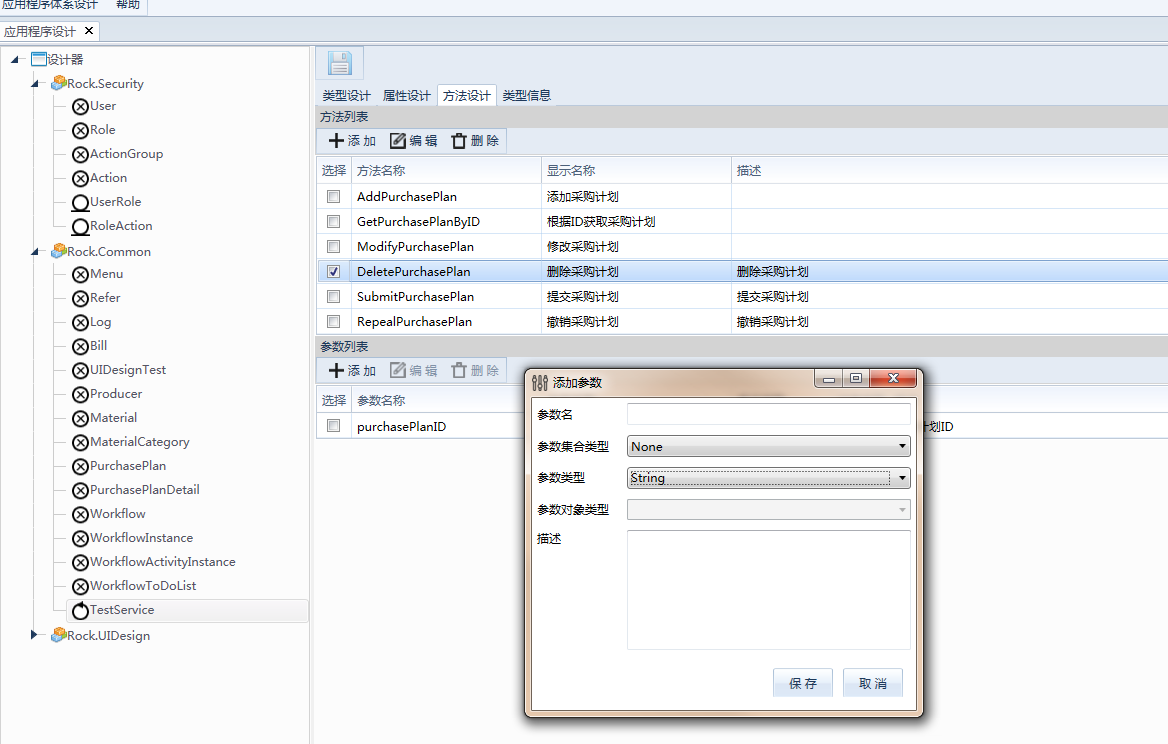
接口参数的维护

选定具体的接口,点击下面的添加按钮就会弹出添加接口参数的页面

参数名称:输入具体的参数名称

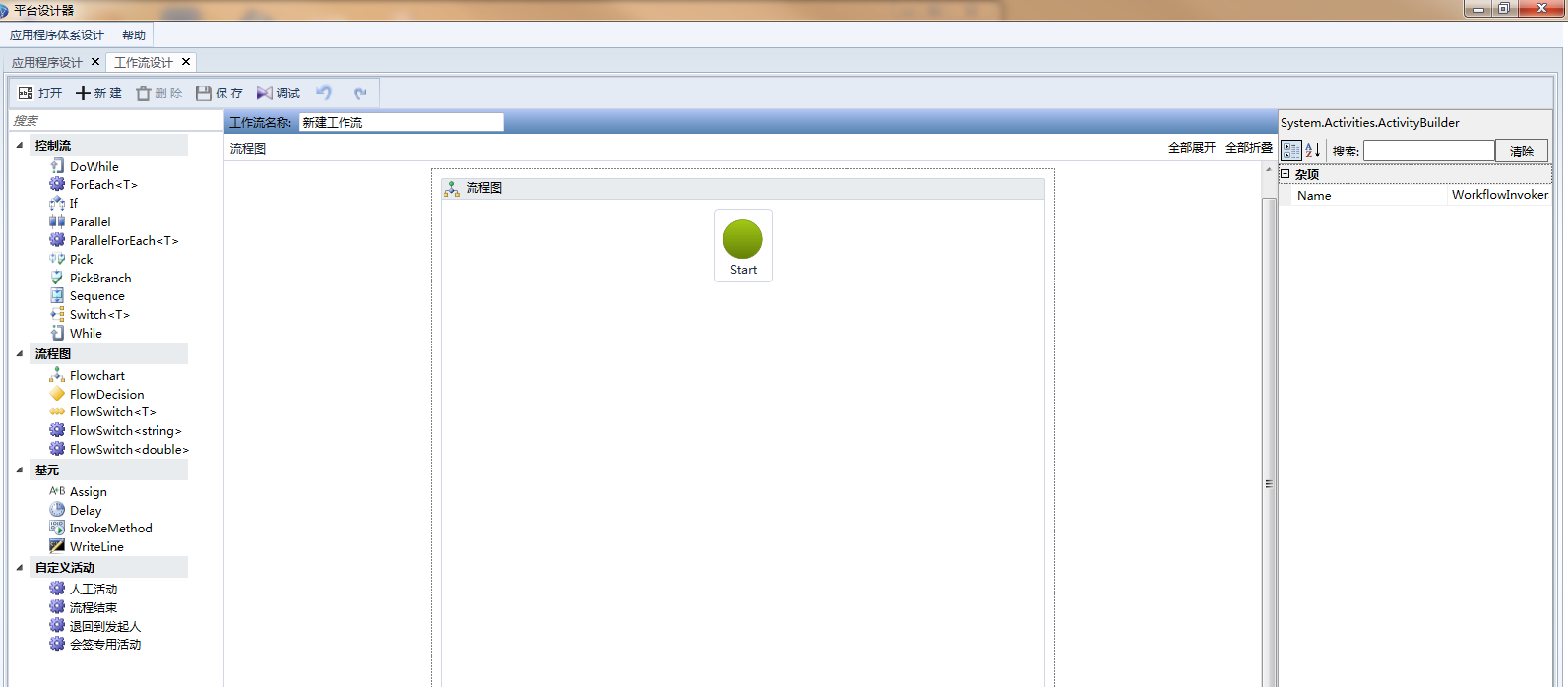
参数集合类型,如果是集合类型则选择List不是则选择None

参数类型:具体的参数类型除了Void和Struct之外都是值类型, 如果选择了Struct,就需要选择参数对象类型.

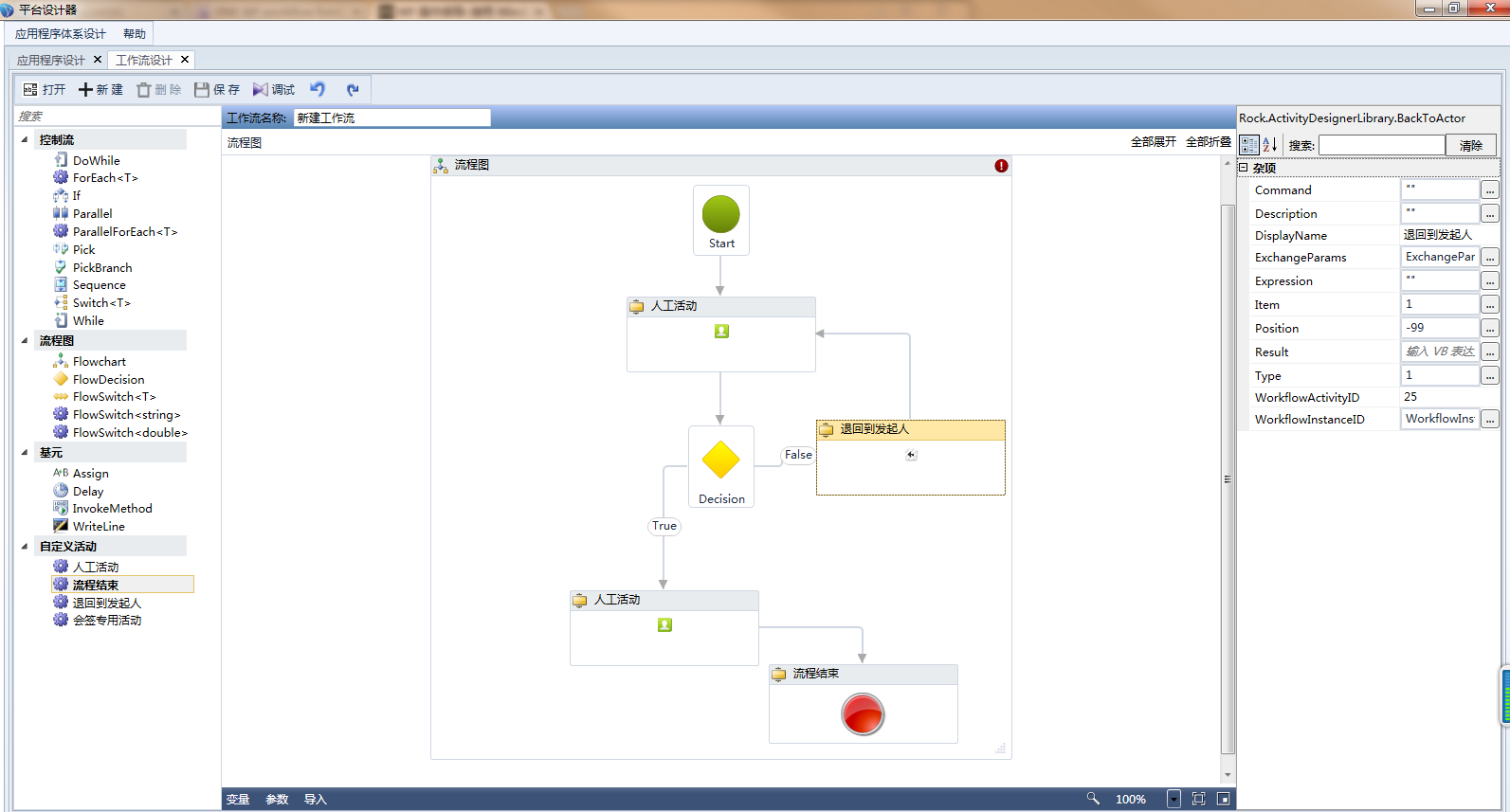


## 工作流设计器

工作流设计器的内核采用的是微软Windows Workflow Foundation框架,采用了一部分微软系统自己的流程控制活动,同时又通过对现实中的业务流程进行研究和抽象,编写了一组自定义的活动,用于系统的业务流程的处理.



流程设计完成后系统有对应的流程引擎进行解析和执行,同时可以通过编写自定义活动支持各种不同类型的流程流转.



# 界面设计

## 简述

目前平台支持以下几种抽象出的表单类型:

* 字典页面模型
* 单据列表页面模型
* 单据维护页面模型
* 树形结构对象页面模型

根据实际的业务情况平台还可以支持更多的表单类型,提供了良好的扩展性。目前平台已经完成的是web页面的页面模型模板,以后会逐步实现Winfom,移动设备的页面模板及解析引擎

### 字典页面模型设计

字典页面设计:



点击页面添加按钮弹出字典页面的基本维护信息

字典页面名称:用于标记字典页面模型的名称

页面类型:页面模型的具体类型名称

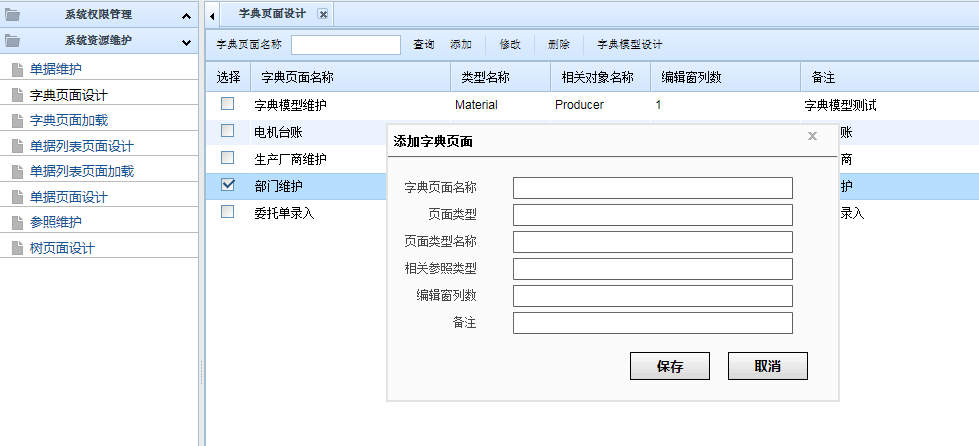
页面类型名称:页面模型类型的中文名称

相关参照类型:在页面类型维护的过程中用到的相关的对象类型,生成的代码中需要加载相应的对象类型,如果没有可以不填

编辑窗列数:用于页面布局根据表单项的多少确定,表单项在编辑页面中分成几栏进行显示,目前平台仅支持1-2栏,如果有需要可以扩展

备注:描述信息

录入以上信息后点击保存按钮,系统会将基本的页面模型存入数据库



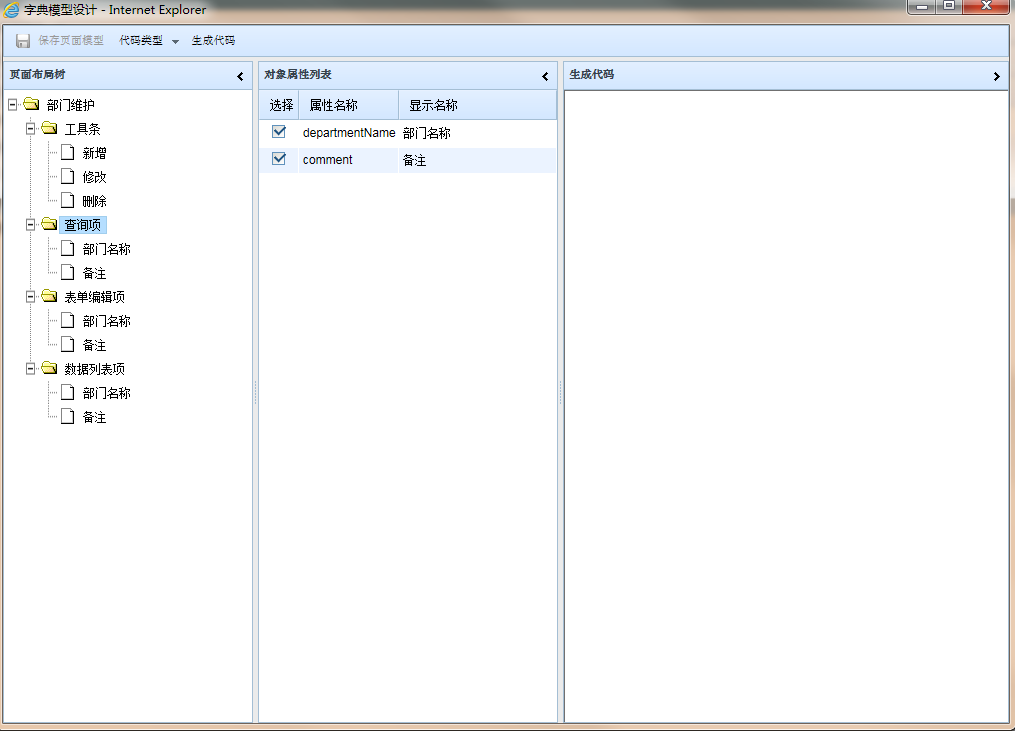
点击字典模型设计按钮,会打开一个新的弹窗,进行页面模型的详细设计

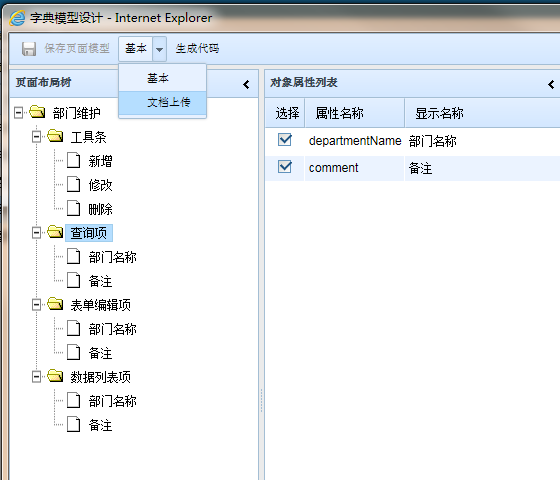
设计器的最左边是模型的页面布局树,根据字典模型的抽象,模型共分为四个部分:

* 工具条:工具条的功能是根据页面模型的类型自动固化的,不可以进行调整
* 查询项:表单页面用于作为查询条件的查询项集合,模板解析引擎会根据在应用程序体系设计中设计的对象属性的信息进行解析生成对应的页面布局及后台的功能代码,查询项的名称叫:QueryItem
* 表单编辑项:表单页面维护时用户录入相关信息的表单项集合, 模板解析引擎会根据在应用程序体系设计中设计的对象属性的信息进行解析生成对应的页面布局及后台的功能代码,查询项的名称叫:FormItem
* 数据列表项:表单页面中表格中显示的列的信息集合, 模板解析引擎会根据在应用程序体系设计中设计的对象属性的信息进行解析生成对应的页面布局及后台的功能代码,列表项的名称叫:GridColumn

字典页面模型支持的模板有两个,一个是基本的无需上传附件的字典对象维护,一个是需要上传附件的字典对象维护,生成代码时可以根据业务实际需要进行选择,同时平台还可以根据需要扩展更多功能类型的维护模板,以适应更多的业务功能需求,并提供了平台良好的扩展性.

选择好代码类型后点击生成代码按钮就会生成表单页面所需的前台HTML和后台Javascript代码.生成的代码还可以根据业务的实际需要进行微调.





### 单据列表页面模型设计

单据类型的表单由于其结构比较复杂,在一个页面无法完成列表选择和具体单据的维护,所以就对单据的维护模型分割成了两部分,一部分是用于查询和导航的单据列表页面模型,另一部分是单据本身的维护页面模型,同时对单据模型的组成部分进行了抽象和定义

一个单据模型包括以下组成部分:

* 主表:主表信息
* 明细表:明细表信息
* 明细表主参照:明细表一般都会引用一个一对多的关系的参照作为明细表的主参照,主参照只能选择一个,例如采购计划单据,主表是采购计划主表,提供采购计划的表头信息,明细表是一个列表表示采购计划的明细,采用计划的内容比如是物料,那么物料就作为明细表的主参照信息.



点击页面添加按钮弹出单据列表页面的基本维护信息

单据列表页面名称:用于标记单据列表页面模型的名称

主表类型:单据的主表类型名称

明细表类型:明细表类型的名称

单据名称:单据类型的中文名称

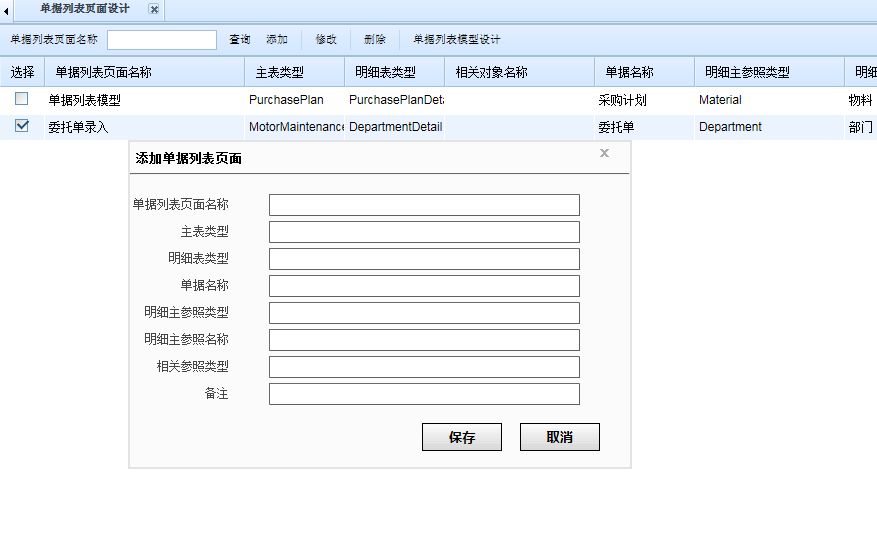
明细表主参照类型:明细表的主参照类型

明细主参照名称:明细表主参照的中文名称

相关参照类型:在页面类型维护的过程中用到的相关的对象类型,生成的代码中需要加载相应的对象类型,如果没有可以不填

备注:描述信息

录入以上信息后点击保存按钮,系统会将基本的单据列表页面模型存入数据库.



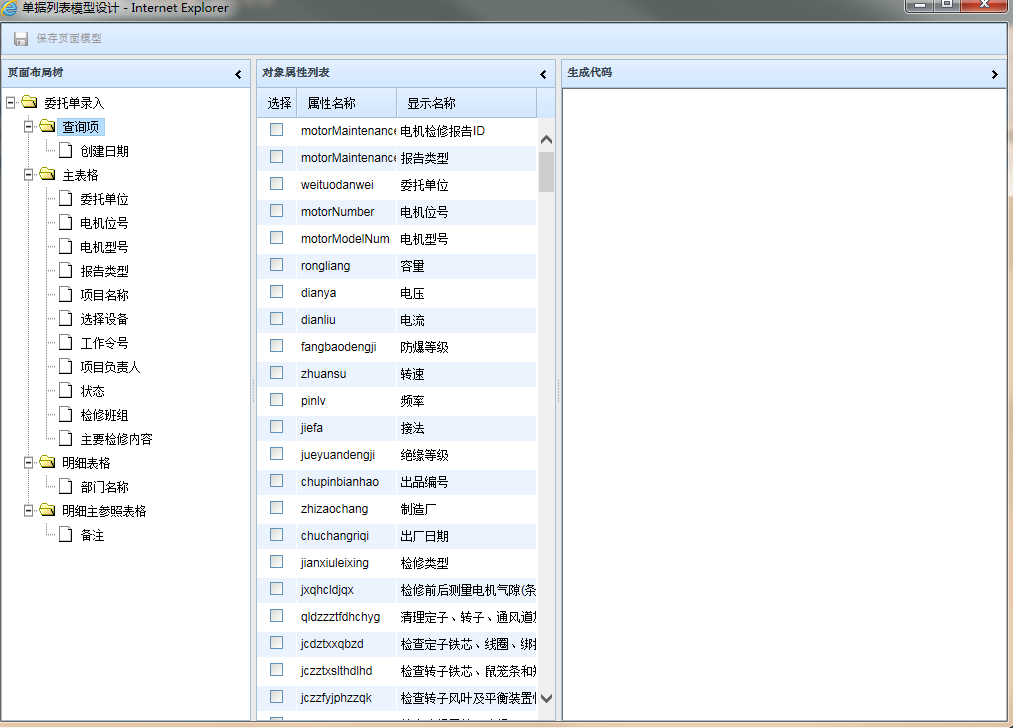
点击单据列表模型设计按钮,会打开一个新的弹窗,进行页面模型的详细设计

设计器的最左边是模型的页面布局树,根据字单据列表模型型的抽象,模型共分为四个部分:

* 查询项:表单页面用于作为查询条件的查询项集合,只能从主表对象的属性中选择,查询项的名称叫:QueryItem
* 主表格:表单页面查询结果的主表的列的集合, 只能从主表对象的属性中选择,表格的名称叫:DataGrid
* 明细表格: 表单页面查询结果的明细表的列的集合, 只能从明细表对象的属性中选择,表格的名称叫:DataGrid
* 明细主参照表格: 表单页面查询结果的明细表的主参照列的集合, 只能从明细表主参照对象的属性中选择,表格的名称叫:DataGrid

**注意:** 单据列表页面上的明细表格中的列事由两部分构成的,一部分是明细表格,另一部分是明细主参照表格中的列,如果明细对象只有主参照的属性,没有属于自己的字段信息明细表格列的集合可以为空,明细主参照表格一般不可以为空

点击保存页面模型按钮,单据列表模型会保存到数据库,并且生成表单页面所需的前台HTML和后台Javascript代码.生成的代码还可以根据业务的实际需要进行微调.



### 单据维护页面模型设计

一个单据列维护页面模型包括以下组成部分:

* 主表:主表信息
* 明细表:明细表信息
* 明细表主参照:明细表一般都会引用一个一对多的关系的参照作为明细表的主参照,主参照只能选择一个,例如采购计划单据,主表是采购计划主表,提供采购计划的表头信息,明细表是一个列表表示采购计划的明细,采用计划的内容比如是物料,那么物料就作为明细表的主参照信息.



点击页面添加按钮弹出单据维护页面的基本维护信息

单据维护页面名称:用于标记单据维护页面模型的名称

主表类型:单据的主表类型名称

明细表类型:明细表类型的名称

单据名称:单据类型的中文名称

明细表主参照类型:明细表的主参照类型

业务服务名称:平台定义的服务契约(接口)名称

明细主参照名称:明细表主参照的中文名称

相关参照类型:在页面类型维护的过程中用到的相关的对象类型,生成的代码中需要加载相应的对象类型,如果没有可以不填

备注:描述信息

录入以上信息后点击保存按钮,系统会将基本的单据维护页面模型存入数据库.



点击单据模型设计按钮,会打开一个新的弹窗,进行页面模型的详细设计

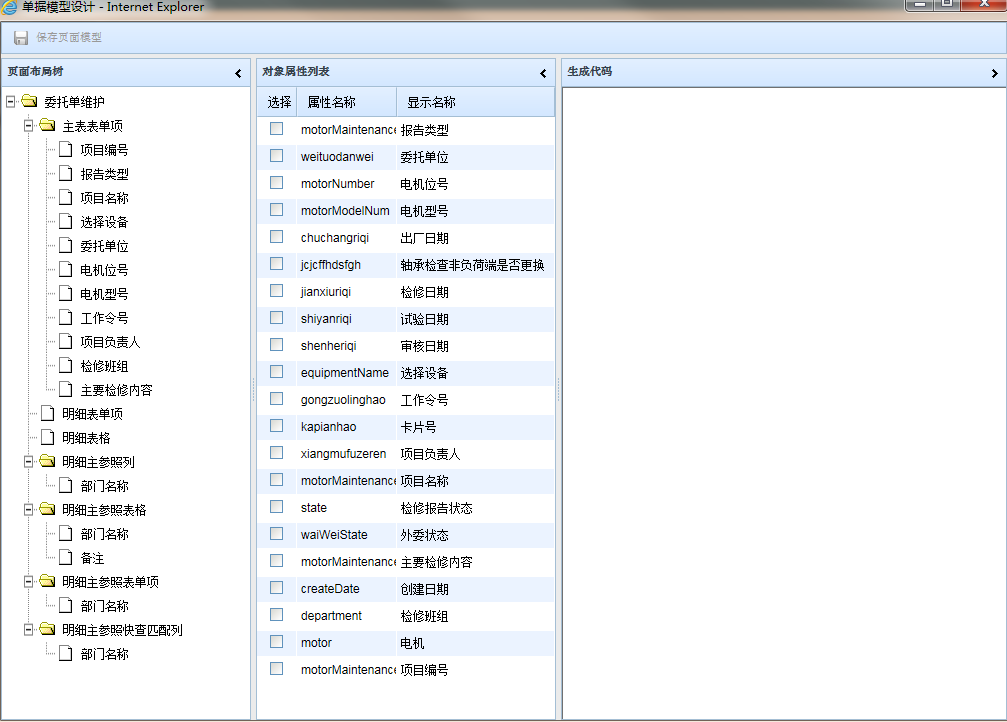
设计器的最左边是模型的页面布局树,根据字单据额维护模型的抽象,模型共分为七个部分:

* 主表表单项:页面主表的编辑表单项集合,只能从主表对象的属性中选择
* 明细表单项:页面添加明细时弹窗中包含的明细表的表单项集合, 只能从明细表对象的属性中选择(弹窗中的表单项由两部分组成: 明细主参照表单项+明细表单项)
* 明细表格:页面明细列表中包含的明细表列的集合, 只能从明细表对象的属性中选择(页面明细视图列表由两部分组成: 明细表格 +明细主参照列)
* 明细主参照列:页面明细列表中需要显示的主参照的列的集合, 只能从明细表主参照对象的属性中选择(明细对象中一般只有明细主参照的对象ID列,如果需要显示明细主参照的其他列就需要在这里添加)
* 明细主参照表格:页面添加明细时弹窗中选择明细弹窗列表时显示的明细主参照列的集合, 只能从明细表主参照对象的属性中选择
* 明细主参照表单项:页面添加明细时弹窗中包含的明细主参照的表单项集合,只能从明细主参照对象的属性中选择
* 明细主参照快查匹配列:页面添加明细弹窗中选择明细主参照对象时,需要用快查的方式快速定位明细主参照对象,这里是进行快速模糊匹配的明细主参照列的集合, 只能从明细表主参照对象的属性中选择

**注意:** 页面明细列表中包含的列是由两部分构成的,一部分是明细表格,另一部分是明细主参照列,如果明细对象只有主参照的属性,没有属于自己的字段信息明细表格列的集合可以为空,明细主参照表格不可以为空

页面添加明细时弹窗中的明细表单项是由两部分组成的,一部分是明细主参照表单项,另一部分是明细表单项,如果明细对象只有主参照的属性,没有属于自己的字段信息,明细表单项的集合可以为空,明细主参照表单项不可以为空

点击保存页面模型按钮,单据维护模型会保存到数据库,并且生成表单页面所需的前台HTML和后台Javascript代码.生成的代码还可以根据业务的实际需要进行微调.



### 树结构页面模型设计

点击页面添加按钮弹出树页面的基本维护信息

树页面名称:用于标记树页面模型的名称

页面类型:页面模型的具体类型名称

页面类型名称:页面模型类型的中文名称

相关参照类型:在页面类型维护的过程中用到的相关的对象类型,生成的代码中需要加载相应的对象类型,如果没有可以不填

编辑窗列数:用于页面布局根据表单项的多少确定,表单项在编辑页面中分成几栏进行显示,目前平台仅支持1-2栏,如果有需要可以扩展

备注:描述信息

录入以上信息后点击保存按钮,系统会将基本的页面模型存入数据库

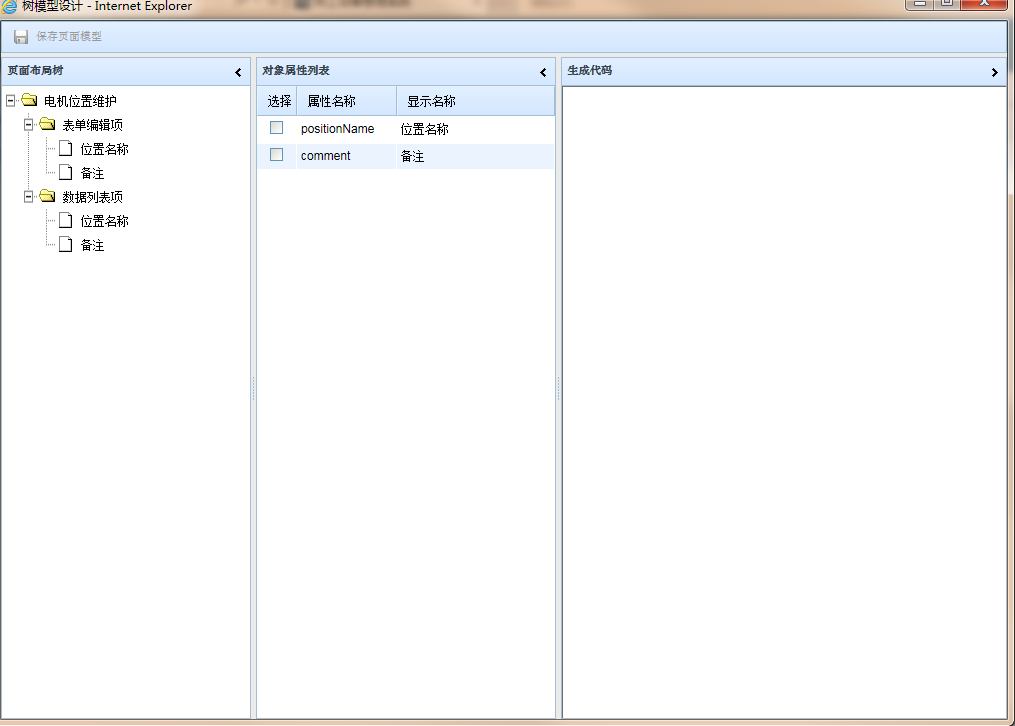


点击树模型设计按钮,会打开一个新的弹窗,进行页面模型的详细设计

设计器的最左边是模型的页面布局树,根据树模型的抽象,模型共分为两个部分:

* 表单编辑项:表单页面维护时用户录入相关信息的表单项集合, 模板解析引擎会根据在应用程序体系设计中设计的对象属性的信息进行解析生成对应的页面布局及后台的功能代码,查询项的名称叫:FormItem
* 数据列表项:表单页面中表格中显示的列的信息集合, 模板解析引擎会根据在应用程序体系设计中设计的对象属性的信息进行解析生成对应的页面布局及后台的功能代码,列表项的名称叫:GridColumn

点击保存页面模型按钮就会生成表单页面所需的前台HTML和后台Javascript代码.生成的代码还可以根据业务的实际需要进行微调.

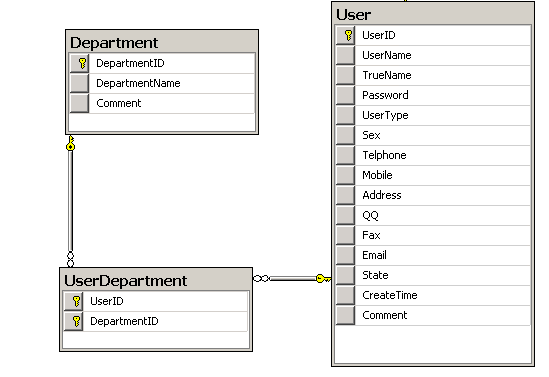


### 多对多关联页面模型设计

一个多对多关联页面模型包括以下组成部分:

* 关联实体:这里的多对多关联实体指的是两个实体之间的多对多关联.
* 主实体:根据实际开发的经验,将多对多关联的两个实体进行划分,一个是主实体,一个是从实体.页面模型要完成主实体的增删改查功能.
* 从实体: 页面模型要完成从实体和主实体之间的关联关系维护.

例如常用的用户和部门的关联关系:部门(Department)为主实体,用户(User)为从实体,一个用户可以属于一个部门,也可以同时属于多个部门,一个部门可以包含一个用户也可以包含多个用户



对多关联页面模型要完成的任务如下:

* 用户对象由单独的页面自行维护.
* 对多关联页面模型完成主实体(Department)的增删改查的功能.
* 对多关联页面模型完成主实体(Department)和从实体(User)之间的多对多关系的建立和移除

点击页面添加按钮弹出添加关联实体页面的基本维护信息

关联实体页面名称:用于标记关联实体页面模型的名称

关联实体类型:关联实体类型名称

主实体类型:主实体类型的名称

从实体类型:从实体类型的名称

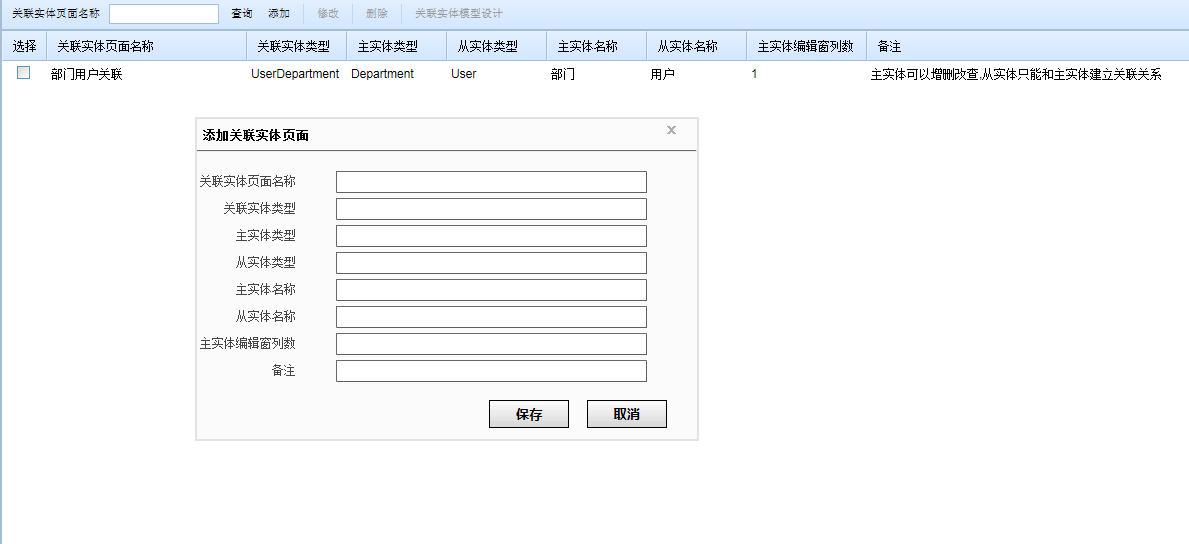
主实体名称:主实体类型的中文名称

从实体名称:从实体类型的中文名称

主实体编辑窗列数:主实体编辑窗页面布局的列数

备注:描述信息

录入以上信息后点击保存按钮,系统会将基本的页面模型存入数据库

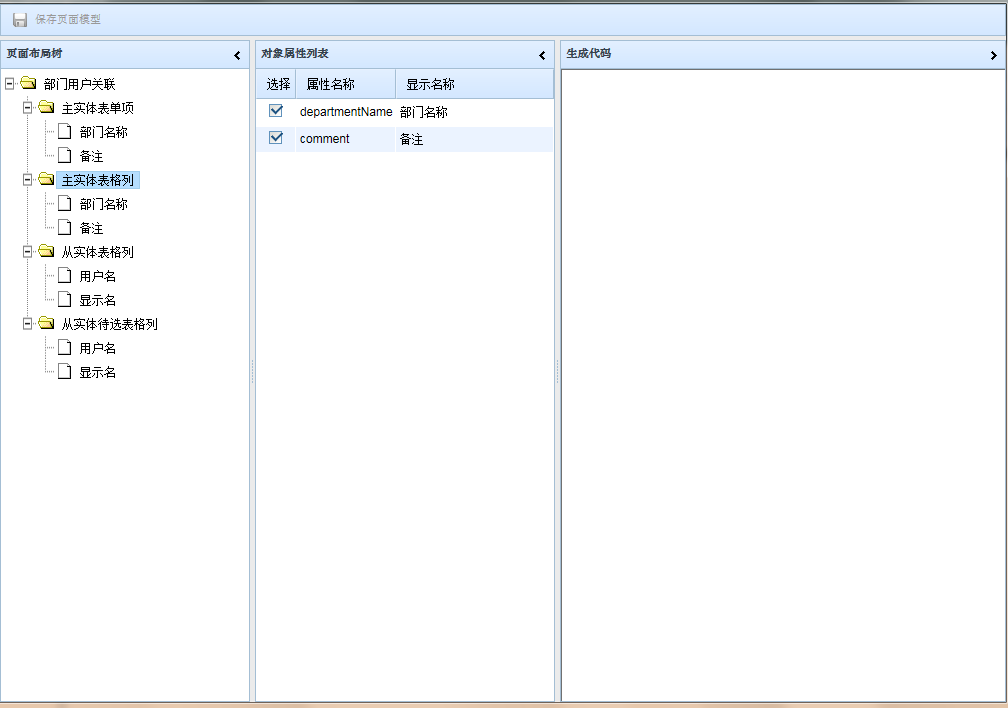


点击树模型设计按钮,会打开一个新的弹窗,进行页面模型的详细设计

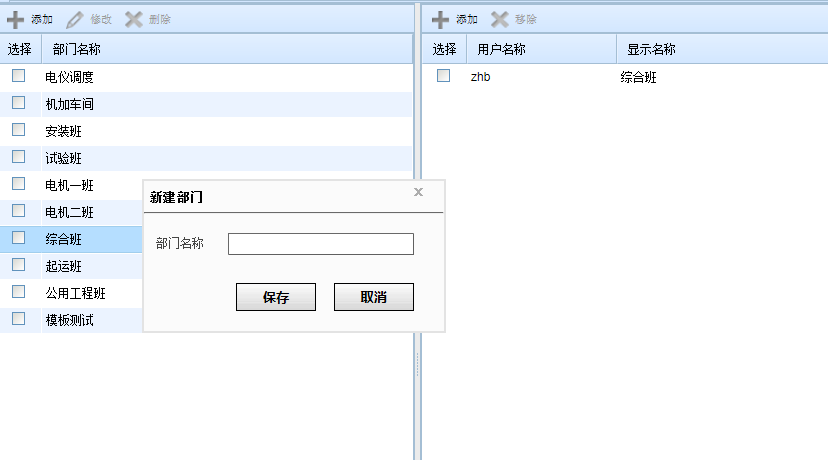
设计器的最左边是模型的页面布局树,根据树模型的抽象,模型共分为四个部分:

* 主表实体表单项:表单页面维护时用户录入主实体相关信息的表单项集合, 模板解析引擎会根据在应用程序体系设计中设计的对象属性的信息进行解析生成对应的页面布局及后台的功能代码,查询项的名称叫:FormItem
* 主实体表格列:表单页面中主实体表格中显示的列的信息集合, 模板解析引擎会根据在应用程序体系设计中设计的对象属性的信息进行解析生成对应的页面布局及后台的功能代码,列表项的名称叫:GridColumn
* 从实体表格列:表单页面中和主实体建立对应关系的从实体表格中显示的列的信息集合, 模板解析引擎会根据在应用程序体系设计中设计的对象属性的信息进行解析生成对应的页面布局及后台的功能代码,列表项的名称叫:GridColumn
* 从实体待选表格列:页面表单中维护主实体和从实体的关联关系时,选择从实体时的弹窗显示的从实体的列信息集合, 模板解析引擎会根据在应用程序体系设计中设计的对象属性的信息进行解析生成对应的页面布局及后台的功能代码,列表项的名称叫:GridColumn

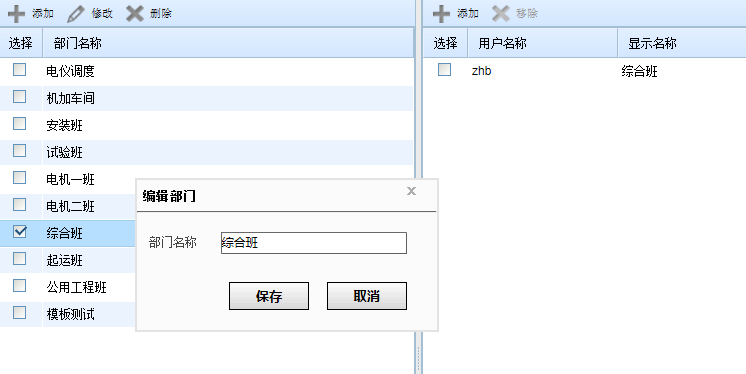
点击保存页面模型按钮就会生成表单页面所需的前台HTML和后台Javascript代码.生成的代码还可以根据业务的实际需要进行微调.



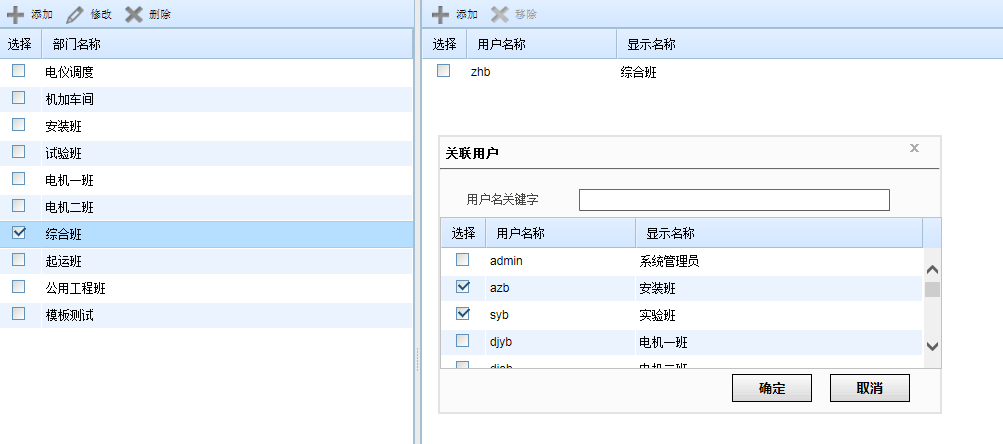
下面的示例就是用户和部门关联关系维护页面生成以后的实际页面效果



新建部门



编辑部门



部门和用户建立和移除关联关系