**Mini框架开发使用**

说明书

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **文件编号** | LZ-FZ-201807 | | | **文件版本** | V1.0 |
| **编写单位** | 青岛励图智能科技有限公司 | | | | |
| **编 写 人** | 张钊 | **编写日期** | 2018-07-05 | | |
| **审 核 人** |  | **审核日期** |  | | |

**修订记录**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **版本** | **修订内容** | **修订人** | **审批人** | **修订日期** |
| V1.0 | 新建文档 |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**目 录**

[1 概述 4](#_Toc2690)

[1.1 文档说明 4](#_Toc16918)

[1.2 框架说明 4](#_Toc5116)

[2 框架介绍 4](#_Toc24149)

[2.1 框架的技术架构 4](#_Toc347)

[2.2 框架的功能架构 5](#_Toc2955)

[2.3 框架层级架构 6](#_Toc26343)

[2.4 框架目录结构说明 6](#_Toc11262)

[2.4.1 前端目录结构 6](#_Toc18537)

[2.4.2 后台目录结构 7](#_Toc7303)

[3 框架初始化 9](#_Toc7873)

[3.1 框架使用开发工具 9](#_Toc27721)

[3.2 框架获取 9](#_Toc22334)

[3.3 获取License 9](#_Toc27430)

[3.4 数据库搭建 9](#_Toc707)

[3.5 系统配置说明 10](#_Toc3292)

[3.5.1 database.config 10](#_Toc23287)

[3.5.2 appSetting.config 11](#_Toc19939)

[3.6 测试搭建情况 11](#_Toc13844)

[3.6.1 测试帐号 11](#_Toc28819)

[3.6.2 功能介绍 12](#_Toc1554)

[4 使用框架开发 13](#_Toc23572)

[4.1 创建helloWord页面 13](#_Toc9790)

[4.1.1 新建模块、界面 13](#_Toc29730)

[4.1.2 添加菜单 13](#_Toc4736)

[4.2 创建我的第一个功能 16](#_Toc14260)

[4.2.1 创建数据库表 16](#_Toc15839)

[4.2.2 仓库信息管理,简单功能 16](#_Toc9724)

[4.2.3 后台开发 22](#_Toc10747)

[4.2.4 首个功能测试 29](#_Toc9317)

[4.3 升级我的第一个功能 34](#_Toc12193)

[4.3.1 查询条件添加 34](#_Toc13684)

[4.3.2 添加字段校验 36](#_Toc10394)

[4.3.3 字典数据的格式化 38](#_Toc4021)

[4.3.4 扩展字典选择字段 42](#_Toc28011)

[4.3.5 使用分页 44](#_Toc23842)

[4.3.6 加入按钮权限 45](#_Toc28067)

[4.3.7 添加操作日志 48](#_Toc28321)

[4.3.8 说明树形结构 48](#_Toc8153)

[5 特定功能 53](#_Toc19593)

[5.1 新建实体类ID 53](#_Toc11936)

[5.2 获取当前用户信息 53](#_Toc5208)

[5.3 添加操作日志 54](#_Toc12778)

[5.4 系统自定义配置 54](#_Toc21750)

[5.5 字典使用说明 54](#_Toc181)

[5.6 数据库切换 59](#_Toc26177)

[5.6.1 SqlServer 59](#_Toc17657)

[5.6.2 MySql 59](#_Toc23979)

[6 技术介绍 59](#_Toc3120)

[6.1 easyUI 59](#_Toc4414)

[6.2 前端集成插件 59](#_Toc8431)

[6.3 树结构层级码 60](#_Toc25405)

[7 系统类库说明 60](#_Toc7778)

[7.1 LT.CoreMini 60](#_Toc9267)

[7.1.1 类库说明 60](#_Toc29878)

[7.1.2 业务相关功能 60](#_Toc1595)

[7.1.3 公共资源 67](#_Toc3502)

[7.2 common.js 67](#_Toc13250)

# 概述

## 文档说明

开发本框架是为了让更多开发人员，通过阅读本文档后能够快速的上手，并进行开发。

## 框架说明

本框架前端使用的是easyui框架,并在其基础上对其样式等进行了封装，后台是自己封装的框架。

# 框架介绍

## 框架的技术架构

框架的技术架构图如下图 2-1所示：



图 2-1 技术架构图

## 框架的功能架构



图 2-2 功能架构图

## 框架层级架构

分别为Handler层、Bll层、Dal层，前台传来请求调用Handler层，在Bll层进行逻辑处理后，调用Dal层进行具体的sql操作,最后返回到前台。

## 框架目录结构说明

### 前端目录结构

前端的目录结构如下图 2-3所示。存放在LT.WebApp目录下的WebAPP文件夹下，WebApp下的文件夹是根据模块名来划分的。例如SysManager指的是系统管理模块，命名方式可以参照数据库表名来命名。

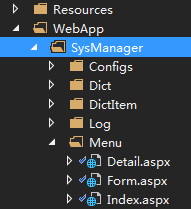


图 2-3 前端目录结构

SysManager目录下的文件夹根据功能来划分，同样可以参照数据库表名来命名，每个功能主要分为三个页面如表 2-1所示：

表 2-1 前端功能页面说明

|  |  |
| --- | --- |
| **功能界面** | **包含功能** |
| Index.aspx | 查询、添加、修改、删除 |
| Form.aspx | 表单添加、修改的提交 |
| Detail.aspx | 数据的详细信息 |

样式、插件、图片等的引用在Resources文件夹，具体说明如表 2-2所示：

表 2-2 引用的说明

|  |  |
| --- | --- |
| **说明** | **存放位置** |
| easyui扩展的一些图片 | Resources/css->defaul->img |
| css引用 | Resources->css |
| js引用 | Resources->js |
| 字体的引用 | Resources->font |

### 后台目录结构

Handler层存放在LT.WebApp目录下的Ashx文件夹下，如图 2-4所示。按照“模块-功能”划分，SysManager指的是系统管理模块。下级是一些功能的请求，可以根据数据库表名来命名，后缀以Handler作为结束，例如数据库表名为Sys\_Menu，Sys指的是模块、Menu指的是功能，可以命名为MenuHandler.aspx。

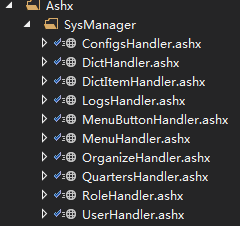


图 2-4 Handler层的目录结构

Bll层存放在LT.Bll目录下，如图 2-5所示。按照“模块-功能”划分，SysManager指的是系统管理模块。下级是一些功能的逻辑处理，可以根据数据库表名来命名，后缀以Bll作为结束，例如数据库表名为Sys\_Menu，Sys指的是模块、Menu指的是功能，可以命名为MenuBll.aspx。

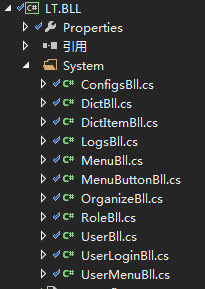


图 2-5 Bll层的目录结构

Dal层存放在Data文件夹下的LT.Dal中，如图 2-6所示。按照“模块-功能”划分，SysManager指的是系统管理模块。下级是一些对数据库的操作，可以根据数据库表名来命名，后缀以Dal作为结束，例如数据库表名为Sys\_Menu，Sys指的是模块、Menu指的是功能，可以命名为MenuDal.aspx。

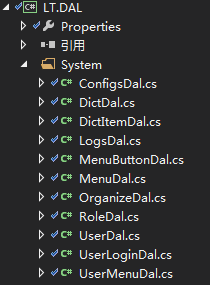


图 2-6 Dal层的目录结构

# 框架初始化

## 框架使用开发工具

下表 3-1为框架使用的开发工具说明：

表 3-1 开发工具说明

|  |  |
| --- | --- |
| **开发工具** | **介绍** |
| Visual Studio 2010+ | VS是一个基本[完整](https://baike.baidu.com/item/%E5%AE%8C%E6%95%B4/32785" \t "https://baike.baidu.com/item/Microsoft%20Visual%20Studio/_blank)的开发工具集，它包括了整个[软件生命周期](https://baike.baidu.com/item/%E8%BD%AF%E4%BB%B6%E7%94%9F%E5%91%BD%E5%91%A8%E6%9C%9F" \t "https://baike.baidu.com/item/Microsoft%20Visual%20Studio/_blank)中所需要的大部分工具，如UML工具、代码管控工具、[集成开发环境](https://baike.baidu.com/item/%E9%9B%86%E6%88%90%E5%BC%80%E5%8F%91%E7%8E%AF%E5%A2%83" \t "https://baike.baidu.com/item/Microsoft%20Visual%20Studio/_blank)(IDE)等等。 |
| Navicat | 数据库管理工具，专为简化数据库的管理及降低系统管理成本而设。 |
| SVN | 开发时用来管理版本的工具。 |

## 框架获取

SVN路径：svn://119.167.153.8/ltdoc/LTGY-Products/框架实现—.NET

## 获取License

## 数据库搭建

新建一个数据MiniTest，库执行System.sql文件，将数据导入到数据库中，说明如表 3-2所示。

表 3-2 数据库说明

|  |  |
| --- | --- |
| **数据库表** | **说明** |
| Sys\_User | 用户信息表 |
| Sys\_UserLogin | 用户登录的信息表 |
| Sys\_UserRole | 用户角色关联表 |
| Sys\_Role | 角色信息表 |
| Sys\_RoleMenu | 角色菜单关联表 |
| Sys\_Menu | 系统菜单表 |
| Sys\_Organize | 组织架构信息表 |
| Sys\_Logs | 系统日志信息表 |
| Sys\_DictItem | 字典值表 |
| Sys\_Dict | 字典表 |
| Sys\_Configs | 系统配置表 |

## 系统配置说明

### database.config

文件位置在LT.WebAPP的Config文件夹下，sqlServer配置代码如下所示：

|  |
| --- |
| <connectionStrings>  <add name="dbconn" connectionString="Server=.;Uid=sa;Pwd=qazwsx123;  Database=LTWMSDB;Max Pool Size=1000; Pooling=true; Asynchronous Processing=true" providerName="System.Data.SqlClient" />  </connectionStrings> |

参数说明入下表 3-3所示：

表 3-3 参数说明

|  |  |
| --- | --- |
| **参数** | **说明** |
| Uid | 代表着数据库的登录名 |
| Pwd | 代表登录密码 |
| Database | 代表链接的数据库 |
| Pooling | 代表连接池是否启用。  连接池就是有一些与数据库的连接放在里面，建立新的连接时系统会先看里面有没有一样的连接，有就不用新建了。  最好是用连接池，默认情况下也是使用连接池的。 |
| Asynchronous Processing | 如果为true表示使用[异步处理](https://www.baidu.com/s?wd=%E5%BC%82%E6%AD%A5%E5%A4%84%E7%90%86&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://zhidao.baidu.com/question/_blank)操作 |
| providerName | ①providerName="System.Data.SqlClient" ----说明使用的是MSSQLServer数据库。  ②providerName="System.Data.SqlLite" ----说明使用的是SQLLite数据库  ③providerName="System.Data.OracleClient" ----说明使用的是Oracle数据库  或providerName="System.Data.Oracle.DataAccess.Client" ----同上  ④providerName="System.Data.OleDb" ----说明使用的是Access数据库 |

### appSetting.config

文件位置在LT.WebAPP的Config文件夹下，说明如下表 3-4所示：

表 3-4 配置说明

|  |  |  |
| --- | --- | --- |
| **key** | **value** | **说明** |
| LoginProvider | Session | 用户登录信息存储的位置Session or Cookie |
| defaultPassword | 123456 | 系统用户的默认密码 |
| defaultDbKey | dbconn | 默认数据库连接信息配置的key值 |
| dbconnEncrypt | false | 数据库连接字符串是否经过加密处理 |
| systemFlag | thisSystem | 当前系统的标记，在记录日志时会使用到，与字典表的配置要对应起来 |
| logSaveType | db | 志记录的方式：db,file,目前先写这两种方式，如果两种都需要记录的话，直接用逗号隔开即可 |

## 测试搭建情况

### 测试帐号

运行起系统后登录界面如下图 3-1所示，输入测试用户admin，测试密码123123。



图 3-1 登录页面

登录成功后如下图 3-2所示：



图 3-2 系统主界面

### 功能介绍

框架的已开发的功能介绍如下表 3-5所示：

表 3-5 功能介绍

|  |  |
| --- | --- |
| **功能** | **说明** |
| 用户管理 | 除了用户基本信息的查看操作外，还包括密码重置，赋予用户角色的功能。 |
| 角色管理 | 除了觉得基本信息的查看操作外，还包括觉得权限的赋予功能。 |
| 菜单管理 | 对菜单信息，按钮信息等管理。分别负责对系统菜单信息的图标、链接、编号等功能的管理，按钮编码、图标、名称等的管理。 |
| 组织架构 | 主要是对部门和组织等的分级管理。 |
| 系统日志 | 对系统操作的记录 |
| 字典值管理 | 包括字典管理和字典值管理，主要是对一些字典类型的管理，比如仓库类型下包括小仓库、大仓库。仓库类型就属于字典类型，小仓库、大仓库属于字典值。 |

# 使用框架开发

## 创建helloWord页面

### 新建模块、界面

写一个简单的Helloworld页面,在WebAPP文件夹下的SysManager文件夹下创建Test文件夹，在Test下创建Index.aspx（Web窗体文件），在body标签内添加一下代码：

|  |
| --- |
| <p>HelloWorld</p> |

### 添加菜单

首先登录系统，在“系统管理”模块中，找到“菜单管理”功能，点击“添加”按钮，弹出如下图 4-1所示弹出框。

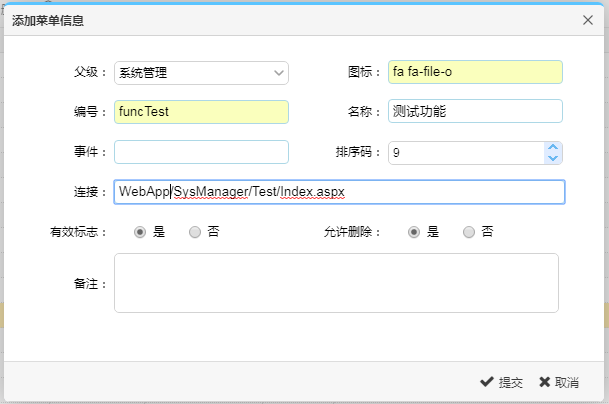


图 4-1 添加菜单信息

添加字段的说明如下表 4-1所示：

表 4-1 字段说明

|  |  |
| --- | --- |
| **字段** | **说明** |
| 父级 | 功能的层级结构 |
| 图标 | 采用的是font awesome的图标库，具体图标查询网站为：http://www.fontawesome.com.cn/faicons。 |
| 编号 | 模块的命名：  分为模块的命名，例如系统管理、基础数据这一类的命名为module+模块名，如moduleSys、moduleBas。  还分功能的命名，是模块下的一个功能，例如用户管理、角色管理，这一类的命名为func+功能的名，例如funcUser、funcRole。以上命名只是参照，并不是规定。 |
| 事件 | 未使用 |
| 排序码 | 数据根据排序码排序 |
| 链接 | 文件对应的路径 |
| 有效标志 | 若为无效，在配置权限中则不显示这条信息 |
| 允许删除 | 是否允许删除这条信息 |

以上信息添加完后点击提交即可。

最后在“系统管理”模块中，找到“角色管理”功能，选中角色编号为“testRole”的这条数据，点击“赋予权限”按钮，弹出如图 4-2所示页面，选中“菜单-测试功能”提交即可。

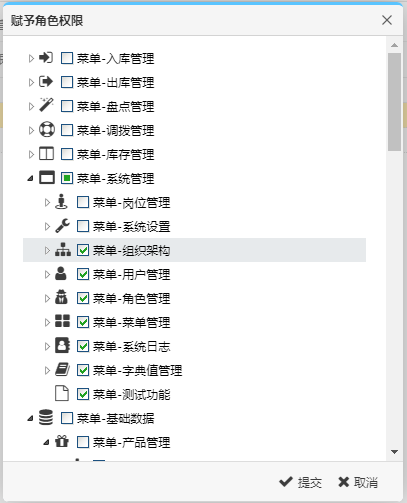


图 4-2 赋予权限弹出框

最后刷新浏览器，就可以看到HelloWorld界面了，如下图 4-3所示：

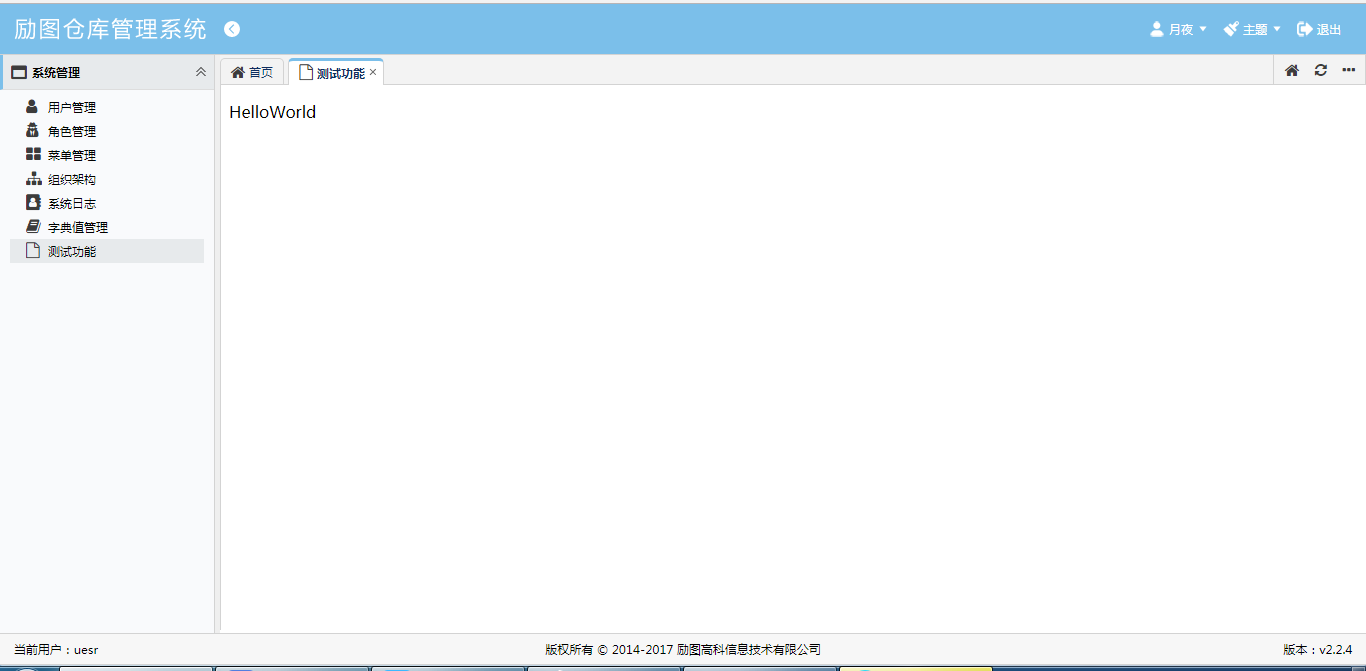


图 4-3 测试页面

## 创建我的第一个功能

### 创建数据库表

在MiniTest数据库中创建Bas\_Storage，代码如下：

|  |
| --- |
| CREATE TABLE [dbo].[Bas\_Storage] (  [F\_Id] VARCHAR (36) NOT NULL,  [F\_Code] VARCHAR (36) NULL,  [F\_Name] nvarchar (36) NULL,  [F\_OrganizeId] VARCHAR (36) NULL,  [F\_Manager] VARCHAR (36) NULL,  [F\_Phone] VARCHAR (36) NULL,  [F\_Address] nvarchar (200) NULL,  [F\_Height] FLOAT (53) NULL,  [F\_Area] FLOAT (53) NULL,  [F\_Type] VARCHAR (36) NULL,  [F\_SortNum] INT NULL,  [F\_DeleteFlag] INT NULL,  [F\_EnabledFlag] INT NULL,  [F\_Description] nvarchar (200) NULL,  [F\_Remark] nvarchar (200) NULL,  [F\_CreateUserId] VARCHAR (36) NULL,  [F\_CreateTime] datetime NULL,  [F\_LastModifyUserId] VARCHAR (36) NULL,  [F\_LastModifyTime] datetime NULL,  [F\_DeleteUserId] VARCHAR (36) NULL,  [F\_DeleteTime] datetime NULL,  [F\_Field1] INT NULL,  [F\_Field2] nvarchar (36) NULL,  [F\_Field3] VARCHAR (36) NULL,  [F\_Field4] datetime NULL  ) |

在这个表中插入以下两条数据。

|  |
| --- |
| INSERT INTO Bas\_Storage  (F\_Id,F\_Code,F\_Name, F\_OrganizeId, F\_Manager, F\_Phone, F\_Address, F\_Height, F\_Area, F\_Type, F\_SortNum, F\_DeleteFlag, F\_EnabledFlag, F\_Description, F\_Remark, F\_CreateUserId, F\_CreateTime, F\_LastModifyUserId, F\_LastModifyTime, F\_DeleteUserId, F\_DeleteTime)  VALUES  ('1','fruitVegetableStorage','果蔬仓', 'f137761a-99f6-4ac8-884e-dd61f8dec574', 'c2ef1427-08a2-4f13-84e5-bec5c1d6c5c2', null, null, '1', '1', 'storageSmall', '1', '0', '1', null, null, 'admin', '2018-04-02 16:01:57.900', 'admin', '2018-04-23 09:54:20.597', null, null) |
| INSERT INTO Bas\_Storage  (F\_Id,F\_Code,F\_Name, F\_OrganizeId, F\_Manager, F\_Phone, F\_Address, F\_Height, F\_Area, F\_Type, F\_SortNum, F\_DeleteFlag, F\_EnabledFlag, F\_Description, F\_Remark, F\_CreateUserId, F\_CreateTime, F\_LastModifyUserId, F\_LastModifyTime, F\_DeleteUserId, F\_DeleteTime)  VALUES  ('2','fruitVegetableStorage','饮品仓', 'f137761a-99f6-4ac8-884e-dd61f8dec574', 'c2ef1427-08a2-4f13-84e5-bec5c1d6c5c2', null, null, '1', '1', 'bigSmall', '1', '0', '1', null, null, 'admin', '2018-04-02 16:01:57.900', 'admin', '2018-04-23 09:54:20.597', null, null) |

### 仓库信息管理,简单功能

在WebAPP下的创建BaseData(这里根据数据库表名来命名，但并不是规定)文件夹，在BaseData文件夹下创建Storage（这里根据数据库表名来命名）文件夹，在Storage文件夹下新建Index.aspx页面。

首先引入js/css等样式文件，将代码放在head标签内代码如下所示：

|  |
| --- |
| <!-- jquery -->  <%-- jquery文件--%>  <script src="../../../Resources/js/jquery/jquery-1.11.3.min.js" type="text/javascript"></script>  <!-- easyUI -->  <%--easyui的icon的样式引入--%>  <link href="../../../Resources/js/easyui/themes/icon.css" rel="stylesheet" type="text/css" />  <%--easyui的样式引入--%>  <link href="../../../Resources/js/easyui/themes/default/easyui.css" rel="stylesheet" type="text/css" />  <%--easyui的扩展样式引入--%>  <link href="../../../Resources/css/easyuiExt.css" rel="stylesheet" type="text/css" />  <%--easyui的js文件引入--%>  <script src="../../../Resources/js/easyui/jquery.easyui.min.js" type="text/javascript"></script>  <%-- easyui的中文转化的引入--%>  <script src="../../../Resources/js/easyui/locale/easyui-lang-zh\_CN.js" type="text/javascript"></script>  <!-- commonJs -->  <%-- 公共样式的引入--%>  <link href="../../../Resources/css/common.css" rel="stylesheet" type="text/css" />  <%--awesome字体的样式引入--%>  <link href="../../../Resources/css/font-awesome.min.css" rel="stylesheet" type="text/css" />  <%---公共方法的引入--%>  <script src="../../../Resources/js/custom/common.js" type="text/javascript"></script> |

body标签修改为<body class="easyui-layout">

在body标签中添加HTML代码，代码如下所示：

|  |
| --- |
| <div region="north" style="overflow: hidden; height: 40px; border-bottom-color: #e5e5e5">  <div class="queryPanel">  <a id="btnQuery" class="easyui-linkbutton"><span class="fa fa-search"></span>查询</a>  </div>  </div>  <div region="center" data-options="border:false">  <div id="gridStorage-tb">  <a class="easyui-linkbutton"><span class="fa fa-plus"></span>添加</a>  <a class="easyui-linkbutton"><span class="fa fa-pencil"></span>修改</a>  <a class="easyui-linkbutton"><span class="fa fa-remove"></span>删除</a>  </div>  <table id="gridStorage" style="table-layout: fixed;"></table>  </div> |

在Script标签中添加如下代码，查询采用的是easyui的datagrid控件详情请查阅easyui的API（使用说明）。

|  |
| --- |
| <script type="text/javascript">  var basePath = "../../../";  var $gridList = $("#gridStorage");  $(function () {  //查询方法  $gridList.datagrid({  url: basePath + 'Ashx/BaseData/StorageHandler.ashx?action=FindList',  toolbar: '#gridStorage-tb',  columns: [[  { title: '编号', field: 'F\_Code', width: 100, align: 'center' },  { title: '名称', field: 'F\_Name', width: 120, align: 'center' },  { title: '部门', field: 'F\_OrganizeId', width: 120, align: 'center' },  { title: '库管员', field: 'F\_Manager', width: 120, align: 'center' },  { title: '联系方式', field: 'F\_Phone', width: 100, align: 'center' },  { title: '仓库高度(m)', field: 'F\_Height', width: 120, align: 'center' },  { title: '仓库面积(㎡)', field: 'F\_Area', width: 120, align: 'center' },  {  title: '仓库类型', field: 'F\_Type', width: 80, align: 'center',  },  {  title: '有效标志', field: 'F\_EnabledFlag', width: 100, align: 'center',  formatter: function (value, row, index) {  return value == 1 ? '<i class="fa fa-toggle-on"></i>' : '<i class="fa fa-toggle-off"></i>';  }  },  { title: '排序码', field: 'F\_SortNum', width: 60, align: 'center' }  ]]  });  //查询的操作  $('#btnQuery').click(function (event) {  $gridList.datagrid('reload');  });  //添加  $('#btnAdd').click(function (event) {  openFullWindow({  id: 'AddFormStorage',  title: '添加仓库信息',  url: 'WebApp/Test/Storage/Form.aspx',  wintype: 'add',  width: 680,  height: 500,  close: function (id, frameId) {  $gridList.datagrid('reload');  }  });  });  //修改按钮的操作  $('#btnEdit').click(function (event) {  var selectItem = $gridList.treegrid('getSelected');  if (!selectItem) {  alertWarnning("请选择要进行修改项！");  return;  }  openFullWindow({  id: "EditFormStorage",  title: '修改仓库信息',  url: 'WebApp/Test/Storage/Form.aspx?fId=' + selectItem.F\_Id,  wintype: 'edit',  width: 680,  height: 500,  close: function (id, frameId) {  $gridList.datagrid('reload');  }  });  });  //删除按钮操作  $('#btnDelete').click(function (event) {  var selectItem = $gridList.datagrid('getSelected');  if (!selectItem) {  alertWarnning("请选择要进行删除的数据！");  return;  }  if (selectItem.F\_EnableDelete == 0) {  alertWarnning("该数据当前不允许删除！");  return;  }  confirm("确定要删除选中数据？", function (e) {  if (e) {  doAjax({  url: basePath + 'Ashx/BaseData/StorageHandler.ashx',  action: "DeleteForm",  data: {  'fId': selectItem.F\_Id  },  success: function (msg) {  alertSuccess(msg);  $gridList.datagrid('reload');  }  });  };  });  });  });  </script> |

在Webapp->BaseData->Storage下新建Form.aspx窗体文件。在head标签下添加以下代码：

|  |
| --- |
| <!-- jquery -->  <%-- jquery文件--%>  <script src="../../../Resources/js/jquery/jquery-1.11.3.min.js" type="text/javascript"></script>  <!-- easyUI -->  <%--easyui的icon的样式引入--%>  <link href="../../../Resources/js/easyui/themes/icon.css" rel="stylesheet" type="text/css" />  <%--easyui的样式引入--%>  <link href="../../../Resources/js/easyui/themes/default/easyui.css" rel="stylesheet" type="text/css" />  <%--easyui的扩展样式引入--%>  <link href="../../../Resources/css/easyuiExt.css" rel="stylesheet" type="text/css" />  <%--easyui的js文件引入--%>  <script src="../../../Resources/js/easyui/jquery.easyui.min.js" type="text/javascript"></script>  <%-- easyui的中文转化的引入--%>  <script src="../../../Resources/js/easyui/locale/easyui-lang-zh\_CN.js" type="text/javascript"></script>  <!-- commonJs -->  <%-- 公共样式的引入--%>  <link href="../../../Resources/css/common.css" rel="stylesheet" type="text/css" />  <%--awesome字体的样式引入--%>  <link href="../../../Resources/css/font-awesome.min.css" rel="stylesheet" type="text/css" />  <%---公共方法的引入--%>  <script src="../../../Resources/js/custom/common.js" type="text/javascript"></script> |

将body标签改为，<body class="easyui-layout">，在body标签下添加如下代码：

|  |
| --- |
| <div data-options="region:'center'">  <form id="formStorage" class="formCls" method="post" enctype="multipart/form-data">  <table>  <tr>  <th class="formTitle">编号：</th>  <td class="formValue">  <input name="F\_Code" id="F\_Code" type="text" class="easyui-textbox" data-options="required:true" />  </td>  <th class="formTitle">名称：</th>  <td class="formValue">  <input name="F\_Name" id="F\_Name" type="text" class="easyui-textbox" data-options="required:true" />  </td>  </tr>  <tr>  <th class="formTitle">部门：</th>  <td class="formValue">  <input name="F\_OrganizeId" id="F\_OrganizeId" type="text" />  </td>  <th class="formTitle">库管员：</th>  <td class="formValue">  <input name="F\_Manager" id="F\_Manager" type="text" />  </td>  </tr>  <tr>  <th class="formTitle">仓库高度(m)：</th>  <td class="formValue">  <input name="F\_Height" id="F\_Height" type="text" class="easyui-numberbox" data-options="min:0,precision:1"/>  </td>  <th class="formTitle">仓库面积(㎡)：</th>  <td class="formValue">  <input name="F\_Area" id="F\_Area" type="text" class="easyui-numberbox" data-options="min:0,precision:1"/>  </td>  </tr>  <tr>  <th class="formTitle">仓库类型：</th>  <td class="formValue">  <input name="F\_Type" id="F\_Type" type="text" />  </td>  <th class="formTitle">排序码：</th>  <td class="formValue">  <input id="F\_SortNum" name="F\_SortNum" class="easyui-numberspinner" data-options="required:true,min:0" />  </td>  </tr>  <tr>  <th class="formTitle">联系方式：</th>  <td class="formValue">  <input id="F\_Phone" name="F\_Phone" class="easyui-textbox" data-options="validType:'mobile'" />  </td>  <th class="formTitle">有效标志：</th>  <td class="formValue">  <label for='F\_EnabledFlag\_True'>  <input type="radio" id="F\_EnabledFlag\_True" name="F\_EnabledFlag" value="1" checked="checked" />是  </label>  <label for='F\_EnabledFlag\_False'>  <input type="radio" id="F\_EnabledFlag\_False" name="F\_EnabledFlag" value="0" />否  </label>  </td>  </tr>  <tr>  <th class="formTitle">地址：</th>  <td class="formValue" colspan="3">  <input id="F\_Address" name="F\_Address" class="easyui-textbox" style="width:96%"/>  </td>  </tr>  <tr>  <th class="formTitle" style="height: 5px"></th>  </tr>  <tr>  <th class="formTitle">描述：</th>  <td class="formValue" colspan="3">  <input name="F\_Description" id="F\_Description" type="text" class="easyui-textbox" data-options="height:60,multiline:true" style="width: 96%" />  </td>  </tr>  <tr>  <th class="formTitle" style="height: 5px"></th>  </tr>  <tr>  <th class="formTitle">备注：</th>  <td class="formValue" colspan="3">  <input name="F\_Remark" id="F\_Remark" type="text" class="easyui-textbox" data-options="height:60,multiline:true" style="width: 96%" />  </td>  </tr>  </table>  <input name="F\_Id" id="F\_Id" type="hidden" />  </form>  </div>  <div data-options="region:'south'" class="formBtnsPanel">  <a id="btnSubmit" class="easyui-linkbutton"><span class="fa fa-check"></span>提交</a>  <a id="btnCancel" class="easyui-linkbutton"><span class="fa fa-remove"></span>取消</a>  </div> |

在script标签下添加如下代码：

|  |
| --- |
| <script type="text/javascript">  var $form = $("#formStorage");  var basePath = "../../../";  $(function () {  var winParam = requestParam(); //获取url传过来的值，必须有：wintype,winid字段  var nowForm = $form.ltform({  url: basePath + "Ashx/BaseData/StorageHandler.ashx",  action: 'SubmitForm',  success: function (data) {  alertSuccess(data, function () {  closeWindow(winParam.winid);  });  }  });  //部门信息  $('#F\_OrganizeId').combobox({  url: basePath + 'Ashx/SysManager/OrganizeHandler.ashx?action=FindSelectTree&fType=2',  valueField: 'id',  width: 175,  panelHeight: 80  });  //库管员  $('#F\_Manager').combobox({  url: basePath + 'Ashx/SysManager/UserHandler.ashx?action=FindSelectList',  width: 175,  panelHeight: 80  });  //仓库类型  $('#F\_Type').combobox({  url: basePath + 'Ashx/Common.ashx?action=GetDictItems&dictCode=storageType',  width: 175,  panelHeight: 80  });  //提交按钮的操作  $("#btnSubmit").click(function () {  nowForm.submit();  });  //取消按钮的操作  $("#btnCancel").click(function () {  closeWindow(winParam.winid);  });  //获取仓库的详细信息  if (winParam.wintype == "edit") {  doAjax({  url: basePath + 'Ashx/BaseData/StorageHandler.ashx',  action: "FindForm",  data: {  'fId': winParam.fId  },  success: function (data) {  $form.form('load', data);  }  });  }  });  </script> |

### 后台开发

#### Model

在LT.Model目录下创建BaseData（这里根据数据库表名来命名）文件夹，在BaseData文件夹下创建StorageModel.cs文件(命名参照：功能名+Model的形式)，继承BaseEntity<StorageModel>。根据功能来继承ICreateAuth(添加)，IModifyAuth(修改)，IDeleteAuth(删除)接口。

|  |
| --- |
| public class StorageModel:BaseEntity<StorageModel>, ICreateAuth, IModifyAuth, IDeleteAuth  {  /// <summary>  /// 主键  /// </summary>  public override string F\_Id { get; set; }  /// <summary>  /// 编码  /// </summary>  public string F\_Code { get; set; }  /// <summary>  /// 名称  [DisplayName("名称")]  public string F\_Name { get; set; }  /// <summary>  /// 部门主键  /// </summary>  public string F\_OrganizeId { get; set; }  /// <summary>  /// 库管员  /// </summary>  public string F\_Manager { get; set; }  /// <summary>  /// 联系方式  /// </summary>  public string F\_Phone { get; set; }  /// <summary>  /// 仓库地址  /// </summary>  public string F\_Address { get; set; }  /// <summary>  /// 仓库高度  /// </summary>  public double? F\_Height { get; set; }  /// <summary>  /// 仓库面积  /// </summary>  public double? F\_Area { get; set; }  /// <summary>  /// 仓库类型  /// </summary>  public string F\_Type { get; set; }  /// <summary>  /// 排序码  /// </summary>  public int? F\_SortNum { get; set; }  /// <summary>  /// 删除标志  /// </summary>  public int? F\_DeleteFlag { get; set; }  /// <summary>  /// 有效标志  /// </summary>  public int? F\_EnabledFlag { get; set; }  /// <summary>  /// 描述  /// </summary>  public string F\_Description { get; set; }  /// <summary>  /// 描述  /// </summary>  public string F\_Remark { get; set; }  /// <summary>  /// 创建人  /// </summary>  public string F\_CreateUserId { get; set; }  /// <summary>  /// 创建时间  /// </summary>  public DateTime? F\_CreateTime { get; set; }  /// <summary>  /// 修改人  /// </summary>  public string F\_LastModifyUserId { get; set; }  /// <summary>  /// 修改时间  /// </summary>  public DateTime? F\_LastModifyTime { get; set; }  /// <summary>  /// 删除人  /// </summary>  public string F\_DeleteUserId { get; set; }  /// <summary>  /// 删除时间  /// </summary>  public DateTime? F\_DeleteTime { get; set; }  } |

#### Ashx

在Ashx目录下创建BaseData文件夹，在BaseData文件夹下创建StorageHandler.ashx文件(命名参照功能名+Handler的形式),继承 FormAshx<StorageModel>接口，重写GetBll方法，代码如下所示：

|  |
| --- |
| public class StorageHandler : FormAshx<StorageModel>  {  StorageBll storageBll = new StorageBll();  /// <summary>  /// 获取表单对象  /// </summary>  /// <returns></returns>  public override FormBll<StorageModel> GetBll()  {  return storageBll;  }  } |

#### Bll

在LT.Bll目录下创建BaseData文件夹，在BaseData文件夹下创建StorageBll.cs文件(命名以功能名+Bll的形式),继承 FormBll<StorageModel>接口，重写GetDal方法，代码如下所示：

|  |
| --- |
| public class StorageBll : FormBll<StorageModel>  {  private StorageDal storageDal = new StorageDal();  /// <summary>  /// 获取数据处理层的信息  /// </summary>  /// <returns></returns>  public override FormDal<StorageModel> GetDal()  {  return storageDal;  }  } |

#### Dal

在LT.Dal目录下创建BaseData文件夹，在BaseData文件夹下创建StorageDal.cs文件(命名以功能名+DAl的形式),继承 FormDal<StorageModel>接口，代码如下所示：

|  |
| --- |
| public class StorageDal : FormDal<StorageModel>  {  /// <summary>  /// 获取表格所有信息  /// </summary>  /// <param name="queryParam"></param>  /// <returns></returns>  public override List<StorageModel> FindList(Dictionary<string, string> queryParam)  {  string sql = @"select storage.\* from Bas\_Storage storage where F\_DeleteFlag=0";  List<DbParameter> paramters = new List<DbParameter>();    sql += "order by F\_SortNum";  List<StorageModel> storages = new List<StorageModel>();  using (DbDataReader dr = DbHelper.ExecuteReader(sql, paramters.ToArray()))  {  while (dr.Read())  {  StorageModel storage = ReaderToModel<StorageModel>(dr);  storages.Add(storage);  }  }  return storages;  }  /// <summary>  /// 获取表格所有信息  /// </summary>  /// <param name="queryParam"></param>  /// <param name="pagination"></param>  /// <returns></returns>  public override List<StorageModel> FindPageList(Dictionary<string, string> queryParam, Pagination pagination)  {  return null;  }  /// <summary>  /// 根据Id获取表单数据  /// </summary>  /// <param name="fId">要查询的表单数据的ID信息。</param>  /// <returns></returns>  public override StorageModel FindForm(string fId)  {  string sql = "select \* from Bas\_Storage where F\_Id=@F\_Id";  DbParameter[] parameters = new SqlParameter[]{  new SqlParameter("@F\_Id", SqlDbType.VarChar){ Value = fId }  };  StorageModel storage = null;  using (DbDataReader dr = DbHelper.ExecuteReader(sql, parameters))  {  if (dr.Read())  {  storage = ReaderToModel<StorageModel>(dr);  }  }  return storage;  }  /// <summary>  /// 添加表单数据  /// </summary>  /// <param name="model">表单的对象信息</param>  /// <returns></returns>  public override bool AddForm(StorageModel model)  {  string sql = @"INSERT INTO Bas\_Storage  (F\_Id,F\_Code,F\_Name,F\_OrganizeId,F\_Manager,F\_Phone,F\_Address,F\_Height,F\_Area,F\_Type,F\_SortNum,F\_DeleteFlag,F\_EnabledFlag,F\_Description,F\_Remark,F\_CreateUserId,F\_CreateTime,F\_LastModifyUserId,F\_LastModifyTime,F\_DeleteUserId,F\_DeleteTime)  VALUES  (@F\_Id,@F\_Code,@F\_Name,@F\_OrganizeId,@F\_Manager,@F\_Phone,@F\_Address,@F\_Height,@F\_Area,@F\_Type,@F\_SortNum,@F\_DeleteFlag,@F\_EnabledFlag,@F\_Description,@F\_Remark,@F\_CreateUserId,@F\_CreateTime,@F\_LastModifyUserId,@F\_LastModifyTime,@F\_DeleteUserId,@F\_DeleteTime)";  DbParameter[] parameters = new SqlParameter[]{  new SqlParameter("@F\_Id", SqlDbType.VarChar, 36){ Value = model.F\_Id },  model.F\_Code == null ? new SqlParameter("@F\_Code", SqlDbType.VarChar, 36) { Value = DBNull.Value } : new SqlParameter("@F\_Code", SqlDbType.VarChar, 36) { Value = model.F\_Code },  model.F\_Name == null ? new SqlParameter("@F\_Name", SqlDbType.NVarChar, 72) { Value = DBNull.Value } : new SqlParameter("@F\_Name", SqlDbType.NVarChar, 72) { Value = model.F\_Name },  model.F\_OrganizeId == null ? new SqlParameter("@F\_OrganizeId", SqlDbType.VarChar, 36) { Value = DBNull.Value } : new SqlParameter("@F\_OrganizeId", SqlDbType.VarChar, 36) { Value = model.F\_OrganizeId },  model.F\_Manager == null ? new SqlParameter("@F\_Manager", SqlDbType.VarChar, 36) { Value = DBNull.Value } : new SqlParameter("@F\_Manager", SqlDbType.VarChar, 36) { Value = model.F\_Manager },  model.F\_Phone == null ? new SqlParameter("@F\_Phone", SqlDbType.VarChar, 36) { Value = DBNull.Value } : new SqlParameter("@F\_Phone", SqlDbType.VarChar, 36) { Value = model.F\_Phone },  model.F\_Address == null ? new SqlParameter("@F\_Address", SqlDbType.NVarChar, 400) { Value = DBNull.Value } : new SqlParameter("@F\_Address", SqlDbType.NVarChar, 400) { Value = model.F\_Address },  model.F\_Height == null ? new SqlParameter("@F\_Height", SqlDbType.Float, -1) { Value = DBNull.Value } : new SqlParameter("@F\_Height", SqlDbType.Float, -1) { Value = model.F\_Height },  model.F\_Area == null ? new SqlParameter("@F\_Area", SqlDbType.Float, -1) { Value = DBNull.Value } : new SqlParameter("@F\_Area", SqlDbType.Float, -1) { Value = model.F\_Area },  model.F\_Type == null ? new SqlParameter("@F\_Type", SqlDbType.NVarChar, 36) { Value = DBNull.Value } : new SqlParameter("@F\_Type", SqlDbType.NVarChar, 36) { Value = model.F\_Type },  model.F\_SortNum == null ? new SqlParameter("@F\_SortNum", SqlDbType.Int, -1) { Value = DBNull.Value } : new SqlParameter("@F\_SortNum", SqlDbType.Int, -1) { Value = model.F\_SortNum },  model.F\_DeleteFlag == null ? new SqlParameter("@F\_DeleteFlag", SqlDbType.Int, -1) { Value = DBNull.Value } : new SqlParameter("@F\_DeleteFlag", SqlDbType.Int, -1) { Value = model.F\_DeleteFlag },  model.F\_EnabledFlag == null ? new SqlParameter("@F\_EnabledFlag", SqlDbType.Int, -1) { Value = DBNull.Value } : new SqlParameter("@F\_EnabledFlag", SqlDbType.Int, -1) { Value = model.F\_EnabledFlag },  model.F\_Description == null ? new SqlParameter("@F\_Description", SqlDbType.NVarChar, 400) { Value = DBNull.Value } : new SqlParameter("@F\_Description", SqlDbType.NVarChar, 400) { Value = model.F\_Description },  model.F\_Remark == null ? new SqlParameter("@F\_Remark", SqlDbType.NVarChar, 400) { Value = DBNull.Value } : new SqlParameter("@F\_Remark", SqlDbType.NVarChar, 400) { Value = model.F\_Remark },  model.F\_CreateUserId == null ? new SqlParameter("@F\_CreateUserId", SqlDbType.VarChar, 36) { Value = DBNull.Value } : new SqlParameter("@F\_CreateUserId", SqlDbType.VarChar, 36) { Value = model.F\_CreateUserId },  model.F\_CreateTime == null ? new SqlParameter("@F\_CreateTime", SqlDbType.DateTime, 8) { Value = DBNull.Value } : new SqlParameter("@F\_CreateTime", SqlDbType.DateTime, 8) { Value = model.F\_CreateTime },  model.F\_LastModifyUserId == null ? new SqlParameter("@F\_LastModifyUserId", SqlDbType.VarChar, 36) { Value = DBNull.Value } : new SqlParameter("@F\_LastModifyUserId", SqlDbType.VarChar, 36) { Value = model.F\_LastModifyUserId },  model.F\_LastModifyTime == null ? new SqlParameter("@F\_LastModifyTime", SqlDbType.DateTime, 8) { Value = DBNull.Value } : new SqlParameter("@F\_LastModifyTime", SqlDbType.DateTime, 8) { Value = model.F\_LastModifyTime },  model.F\_DeleteUserId == null ? new SqlParameter("@F\_DeleteUserId", SqlDbType.VarChar, 36) { Value = DBNull.Value } : new SqlParameter("@F\_DeleteUserId", SqlDbType.VarChar, 36) { Value = model.F\_DeleteUserId },  model.F\_DeleteTime == null ? new SqlParameter("@F\_DeleteTime", SqlDbType.DateTime, 8) { Value = DBNull.Value } : new SqlParameter("@F\_DeleteTime", SqlDbType.DateTime, 8) { Value = model.F\_DeleteTime }  };  int line = DbHelper.ExecuteNonQuery(sql, parameters);  return line == 1; }  /// <summary>  /// 修改表单数据  /// </summary>  /// <param name="model">表单的对象信息</param>  /// <returns></returns>  public override bool EditForm(StorageModel model)  {  string sql = @"UPDATE Bas\_Storage SET  F\_Code=@F\_Code,F\_Name=@F\_Name,F\_OrganizeId=@F\_OrganizeId,F\_Manager=@F\_Manager,F\_Phone=@F\_Phone,F\_Address=@F\_Address,F\_Height=@F\_Height,F\_Area=@F\_Area,F\_Type=@F\_Type,F\_SortNum=@F\_SortNum,F\_DeleteFlag=@F\_DeleteFlag,F\_EnabledFlag=@F\_EnabledFlag,F\_Description=@F\_Description,F\_Remark=@F\_Remark,F\_LastModifyUserId=@F\_LastModifyUserId,F\_LastModifyTime=@F\_LastModifyTime,F\_DeleteUserId=@F\_DeleteUserId,F\_DeleteTime=@F\_DeleteTime  WHERE F\_Id=@F\_Id";  DbParameter[] parameters = new SqlParameter[]{  model.F\_Code == null ? new SqlParameter("@F\_Code", SqlDbType.VarChar, 36) { Value = DBNull.Value } : new SqlParameter("@F\_Code", SqlDbType.VarChar, 36) { Value = model.F\_Code },  model.F\_Name == null ? new SqlParameter("@F\_Name", SqlDbType.NVarChar, 72) { Value = DBNull.Value } : new SqlParameter("@F\_Name", SqlDbType.NVarChar, 72) { Value = model.F\_Name },  model.F\_OrganizeId == null ? new SqlParameter("@F\_OrganizeId", SqlDbType.VarChar, 36) { Value = DBNull.Value } : new SqlParameter("@F\_OrganizeId", SqlDbType.VarChar, 36) { Value = model.F\_OrganizeId },  model.F\_Manager == null ? new SqlParameter("@F\_Manager", SqlDbType.VarChar, 36) { Value = DBNull.Value } : new SqlParameter("@F\_Manager", SqlDbType.VarChar, 36) { Value = model.F\_Manager },  model.F\_Phone == null ? new SqlParameter("@F\_Phone", SqlDbType.VarChar, 36) { Value = DBNull.Value } : new SqlParameter("@F\_Phone", SqlDbType.VarChar, 36) { Value = model.F\_Phone },  model.F\_Address == null ? new SqlParameter("@F\_Address", SqlDbType.NVarChar, 400) { Value = DBNull.Value } : new SqlParameter("@F\_Address", SqlDbType.NVarChar, 400) { Value = model.F\_Address },  model.F\_Height == null ? new SqlParameter("@F\_Height", SqlDbType.Float, -1) { Value = DBNull.Value } : new SqlParameter("@F\_Height", SqlDbType.Float, -1) { Value = model.F\_Height },  model.F\_Area == null ? new SqlParameter("@F\_Area", SqlDbType.Float, -1) { Value = DBNull.Value } : new SqlParameter("@F\_Area", SqlDbType.Float, -1) { Value = model.F\_Area },  model.F\_Type == null ? new SqlParameter("@F\_Type", SqlDbType.NVarChar, 36) { Value = DBNull.Value } : new SqlParameter("@F\_Type", SqlDbType.NVarChar, 36) { Value = model.F\_Type },  model.F\_SortNum == null ? new SqlParameter("@F\_SortNum", SqlDbType.Int, -1) { Value = DBNull.Value } : new SqlParameter("@F\_SortNum", SqlDbType.Int, -1) { Value = model.F\_SortNum },  model.F\_DeleteFlag == null ? new SqlParameter("@F\_DeleteFlag", SqlDbType.Int, -1) { Value = DBNull.Value } : new SqlParameter("@F\_DeleteFlag", SqlDbType.Int, -1) { Value = model.F\_DeleteFlag },  model.F\_EnabledFlag == null ? new SqlParameter("@F\_EnabledFlag", SqlDbType.Int, -1) { Value = DBNull.Value } : new SqlParameter("@F\_EnabledFlag", SqlDbType.Int, -1) { Value = model.F\_EnabledFlag },  model.F\_Description == null ? new SqlParameter("@F\_Description", SqlDbType.NVarChar, 400) { Value = DBNull.Value } : new SqlParameter("@F\_Description", SqlDbType.NVarChar, 400) { Value = model.F\_Description },  model.F\_Remark == null ? new SqlParameter("@F\_Remark", SqlDbType.NVarChar, 400) { Value = DBNull.Value } : new SqlParameter("@F\_Remark", SqlDbType.NVarChar, 400) { Value = model.F\_Remark },  model.F\_LastModifyUserId == null ? new SqlParameter("@F\_LastModifyUserId", SqlDbType.VarChar, 36) { Value = DBNull.Value } : new SqlParameter("@F\_LastModifyUserId", SqlDbType.VarChar, 36) { Value = model.F\_LastModifyUserId },  model.F\_LastModifyTime == null ? new SqlParameter("@F\_LastModifyTime", SqlDbType.DateTime, 8) { Value = DBNull.Value } : new SqlParameter("@F\_LastModifyTime", SqlDbType.DateTime, 8) { Value = model.F\_LastModifyTime },  model.F\_DeleteUserId == null ? new SqlParameter("@F\_DeleteUserId", SqlDbType.VarChar, 36) { Value = DBNull.Value } : new SqlParameter("@F\_DeleteUserId", SqlDbType.VarChar, 36) { Value = model.F\_DeleteUserId },  model.F\_DeleteTime == null ? new SqlParameter("@F\_DeleteTime", SqlDbType.DateTime, 8) { Value = DBNull.Value } : new SqlParameter("@F\_DeleteTime", SqlDbType.DateTime, 8) { Value = model.F\_DeleteTime },  new SqlParameter("@F\_Id", SqlDbType.VarChar, 36){ Value = model.F\_Id }  };  int line = DbHelper.ExecuteNonQuery(sql, parameters);  return line == 1;  }  /// <summary>  /// 根据Id删除相应的数据  /// </summary>  /// <param name="fId">要删除的表单ID信息</param>  /// <returns></returns>  public override bool DeleteForm(string fId)  {  string sql = "update Bas\_Storage set F\_DeleteFlag=1 where F\_Id=@F\_Id";  DbParameter[] parameters = new SqlParameter[]{  new SqlParameter("@F\_Id", SqlDbType.VarChar){ Value = fId }  };  int line = DbHelper.ExecuteNonQuery(sql, parameters);  return line == 1;  }  } |

### 首个功能测试

登录系统，找到“系统管理”->“菜单管理”，点击“添加”按钮，弹出如下图 4-4所示的窗口。按照图片上的信息添加“基础数据”菜单信息。

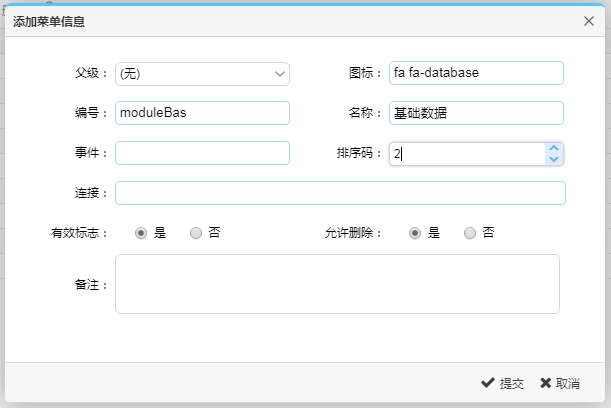


图 4-4 添加菜单信息页面

然后再点击“添加”按钮，弹出如下图 4-5所示的窗口，按照图片上的信息添加“仓库管理”的菜单信息。

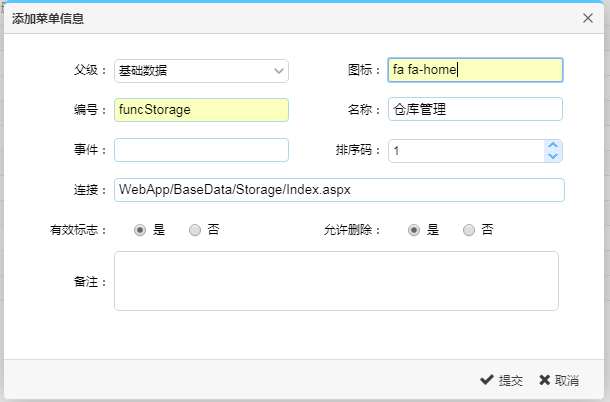


图 4-5 添加菜单信息

打开“系统管理”->“角色管理”，选中角色编号是“testRole”的这条信息，点击“赋予权限”按钮，选中相应的权限如图 4-6所示：

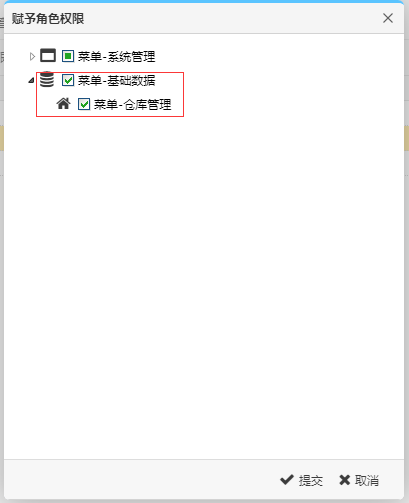


图 4-6 赋予角色权限页面

点击提交后刷新浏览器，点击仓库管理页面如下图 4-7所示，查询出的数据正确。

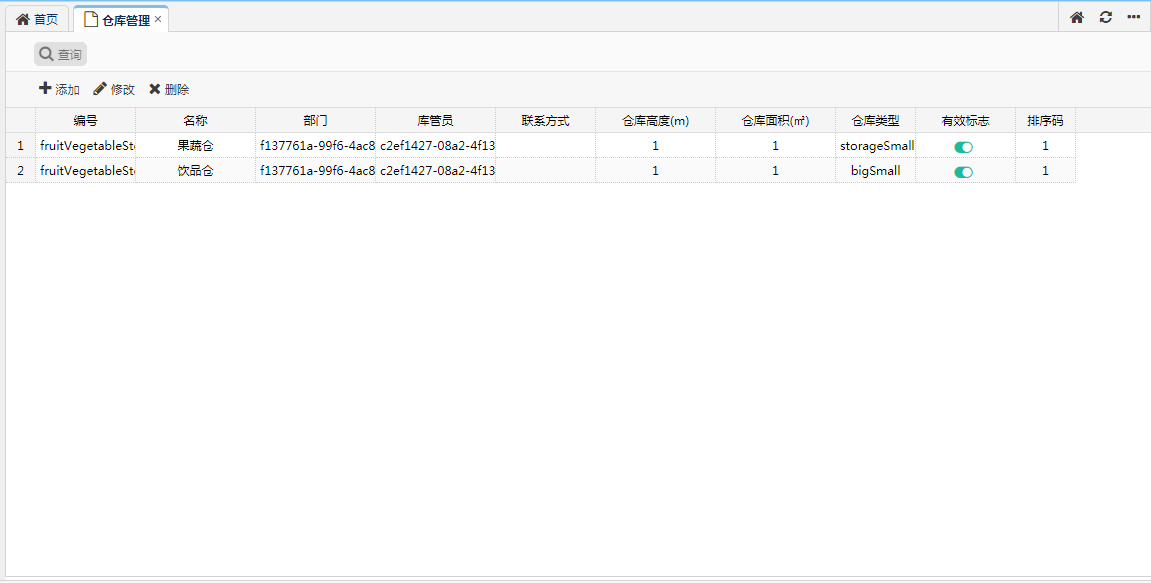


图 4-7 仓库管理界面

选中一条数据点击“删除”按钮，弹出下图 4-8所示对话框，点击确定后数据就会被删除。

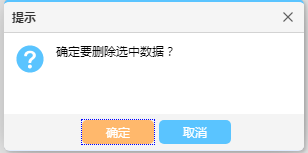


图 4-8 提示框

在“基础数据”->“仓库管理”功能下，点击“添加”按钮，弹出如下图 4-9所示的窗口。

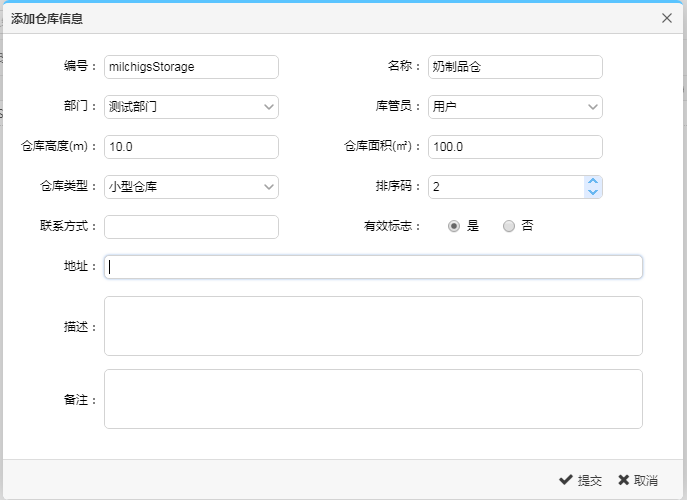


图 4-9 添加仓库信息页面

点击提交后添加一条数据，如图 4-10所示：

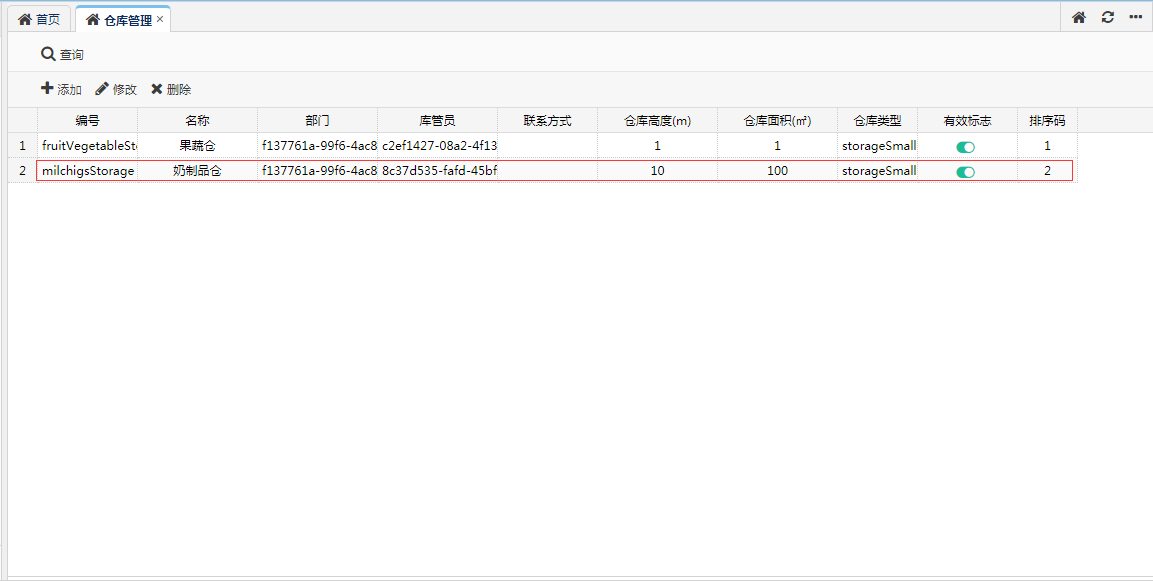


图 4-10 仓库管理页面

选中名称为“奶制品仓”这条数据点击“修改”按钮，弹出如下图 4-11所示的窗口，输入联系方式。

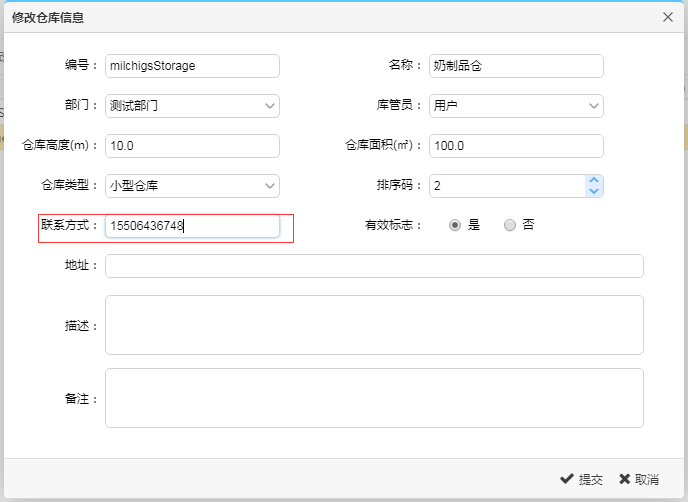


图 4-11 修改仓库信息页面

点击提交按钮后，修改了这条记录如图 4-12所示：

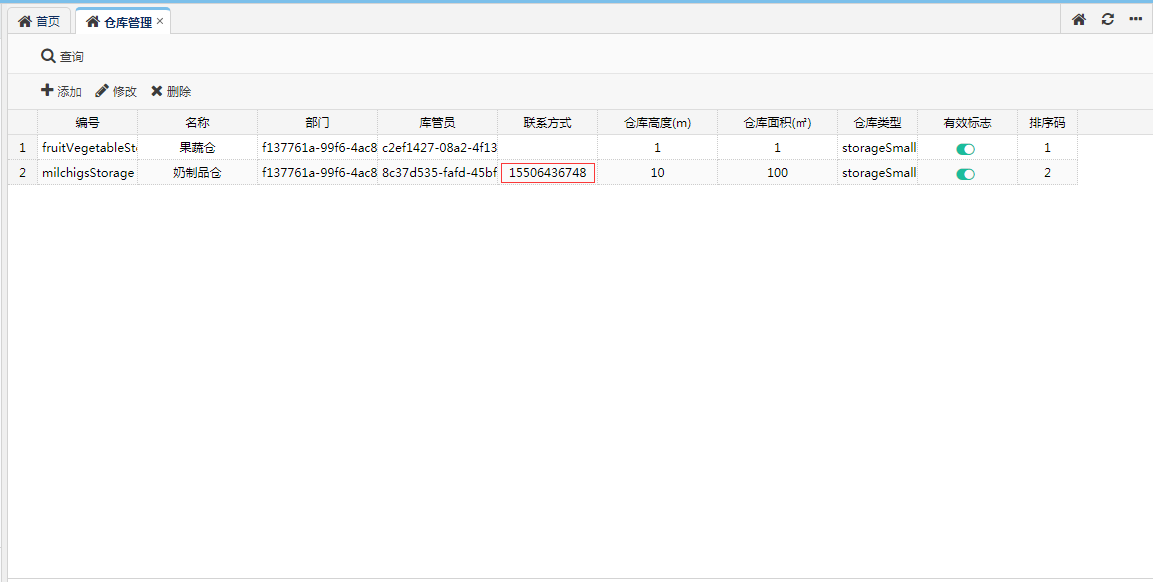


图 4-12 仓库管理页面

## 升级我的第一个功能

### 查询条件添加

在BaseData文件夹下的Storage文件加中，在<div class="queryPanel">的标签内添加以下代码。

|  |
| --- |
| <span>关键词：</span>  <input id="keyword" class="easyui-textbox" prompt="请输入编号或名称"/> |

然后在datagrid控件中添加onBeforeLoad属性(查看easyui使用说明)，传递额外参数给后台，后台根据这个参数筛选出结果。

|  |
| --- |
| $gridList.datagrid({  toolbar: '#gridStorage-tb',  url: basePath + 'Ashx/BaseData/StorageHandler.ashx?action=FindList',  columns: [[  { title: '主键', field: 'F\_Id', hidden: true },  { title: '编号', field: 'F\_Code', width: 100, align: 'center' },  { title: '名称', field: 'F\_Name', width: 120, align: 'center' },  { title: '部门', field: 'F\_OrganizeId', width: 120, align: 'center'},  { title: '库管员', field: 'F\_Manager', width: 120, align: 'center'},  { title: '联系方式', field: 'F\_Phone', width: 100, align: 'center' },  { title: '仓库高度(m)', field: 'F\_Height', width: 120, align: 'center' },  { title: '仓库面积(㎡)', field: 'F\_Area', width: 120, align: 'center' },  {  title: '仓库类型', field: 'F\_Type', width: 80, align: 'center',  },  {  title: '有效标志', field: 'F\_EnabledFlag', width: 100, align: 'center',  formatter: function (value, row, index) {  return value == 1 ? '<i class="fa fa-toggle-on"></i>' : '<i class="fa fa-toggle-off"></i>';  }  },  { title: '排序码', field: 'F\_SortNum', width: 60, align: 'center' }  ]],  onBeforeLoad: function (param) {  param.keyword = $('#keyword').textbox('getValue');  }  }); |

后台LT.Dal->BaseData->StorageDal文件中的FindList方法添加如下代码：

|  |
| --- |
| public override List<StorageModel> FindList(Dictionary<string, string> queryParam)  {  string sql = @"select storage.\* from Bas\_Storage storage where F\_DeleteFlag=0";  List<DbParameter> paramters = new List<DbParameter>();  if (queryParam.ContainsKey("keyword"))  {  sql += "and (storage.F\_Name like @F\_Name or storage.F\_Code like @F\_Code)";  paramters.Add(new SqlParameter("@F\_Code", SqlDbType.NVarChar) { Value = string.Format("%{0}%", queryParam["keyword"]) });  paramters.Add(new SqlParameter("@F\_Name", SqlDbType.NVarChar) { Value = string.Format("%{0}%", queryParam["keyword"]) });  }    sql += "order by F\_SortNum";  List<StorageModel> storages = new List<StorageModel>();  using (DbDataReader dr = DbHelper.ExecuteReader(sql, paramters.ToArray()))  {  while (dr.Read())  {  StorageModel storage = ReaderToModel<StorageModel>(dr);  storages.Add(storage);  }  }  return storages;  } |

以上代码添加成功后，页面展示如图 4-13所示：

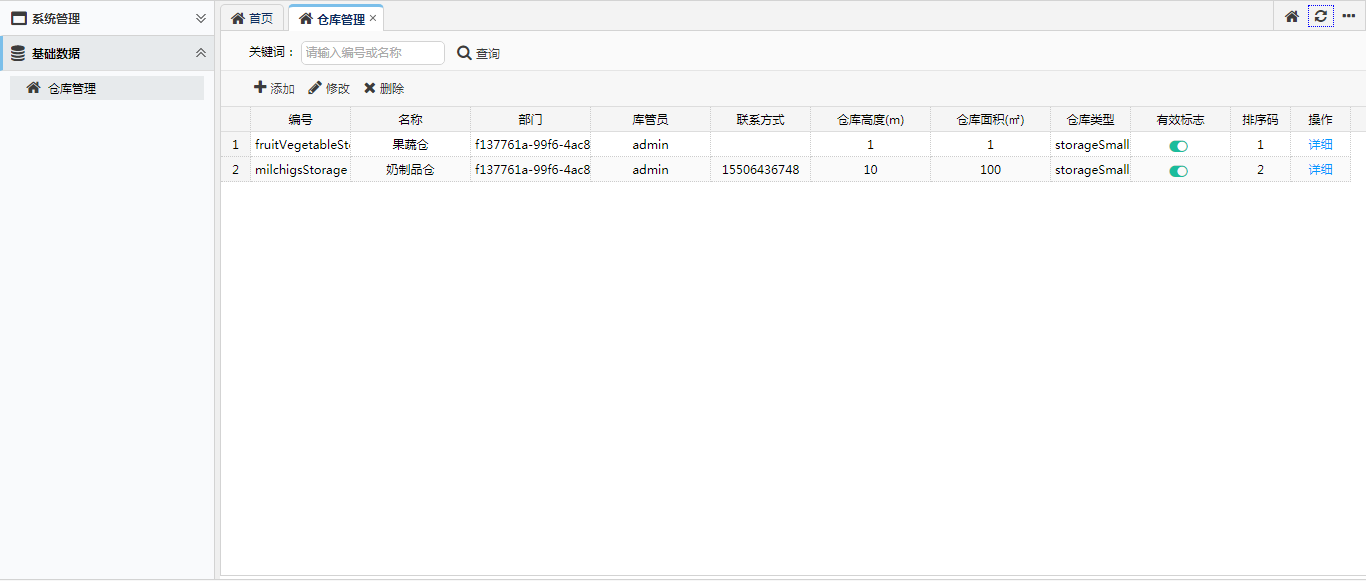


图 4-13 仓库管理页面

关键词中输入例如“fr”关键字，点击查询，查询出的结果如图 4-14所示。

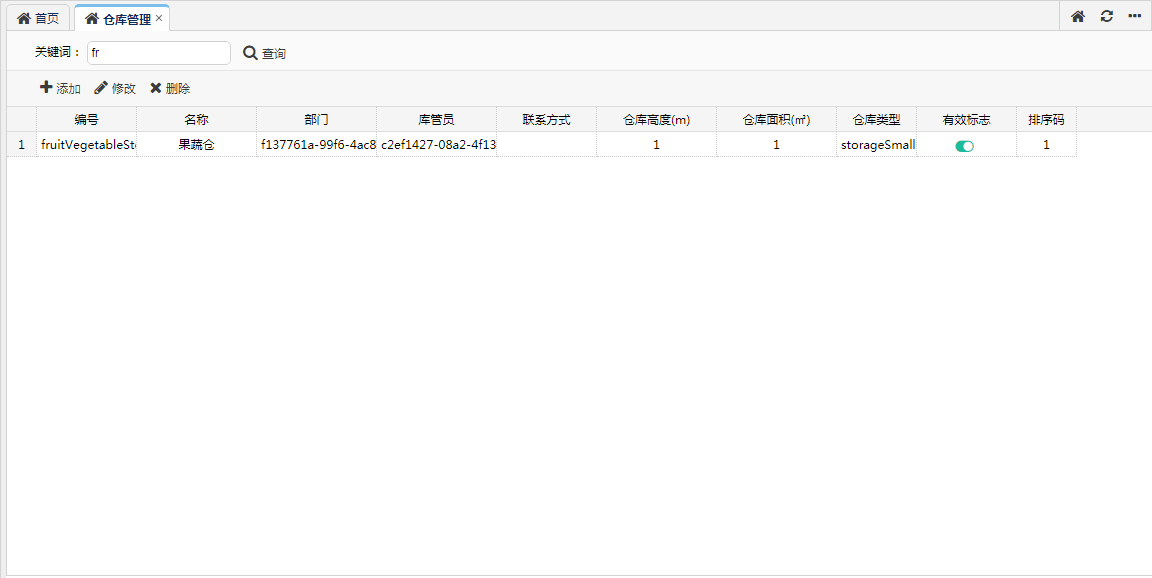


图 4-14 仓库管理页面

### 添加字段校验

在input标签中添加data-options="required:true"，验证必须是输入项，否则提示错误。如WebApp->BaseData->Storage->Form.aspx(仓库的表单页面)中，编号中的验证如下所示：

|  |
| --- |
| <th class="formTitle">编号：</th>  <td class="formValue">  <input name="F\_Code" id="F\_Code" type="text" class="easyui-textbox" data-options="required:true" />  </td> |

如果编号未输入页面如图 4-15所示，输入框为浅红色。

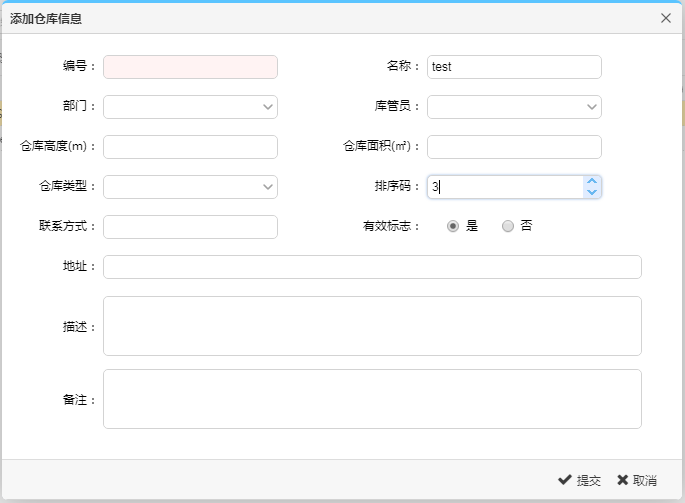


图 4-15 添加仓里信息页面

如果编号没有填写，点击提交，提示如图 4-16所示：

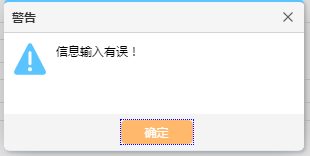


图 4-16 信息提示

验证邮箱则在data-options添加validType:'email'属性。

参照WebApp->SysManager->User->Form.aspx(用户管理的表单)的邮箱字段,代码如下所示：

|  |
| --- |
| <th class="formTitle">电子邮件：</th>  <td class="formValue">  <input name="F\_Email" id="F\_Email" type="text" class="easyui-textbox" data-options="validType:'email'" />  </td> |

如果输入的格式不正确，则显示如图 4-17所示的错误：

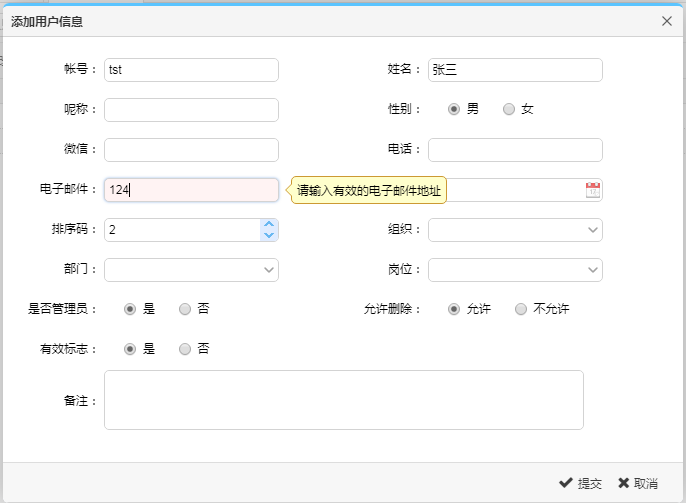


图 4-17 添加用户信息页面

常用的验证如手机验证、字符长度长度验证等。使用方法为在data-options的方法中添加validType:’mobile’、validType:’length[0,20]’，其他的验证可以参考easyui使用手册。

### 字典数据的格式化

在“系统管理”->“字典管理”，点击“字典设置”，弹出如下图 4-18所示的窗口：

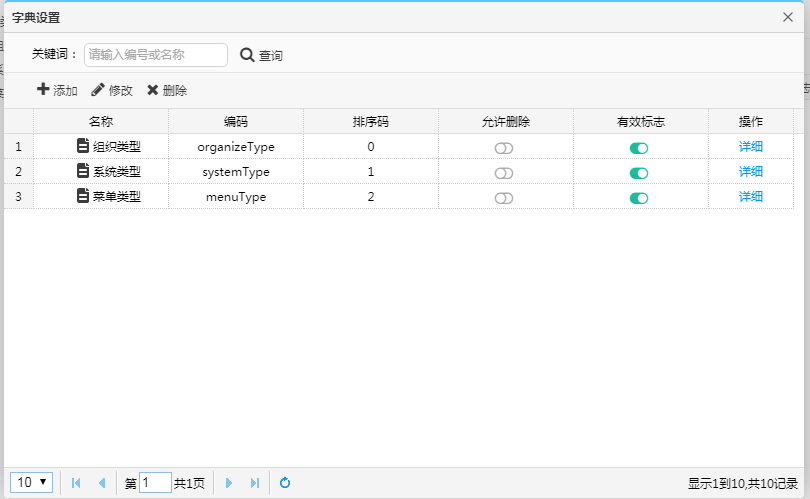


图 4-18 字典设置页面

点击添加按钮，按照下图 4-19所示添加内容。

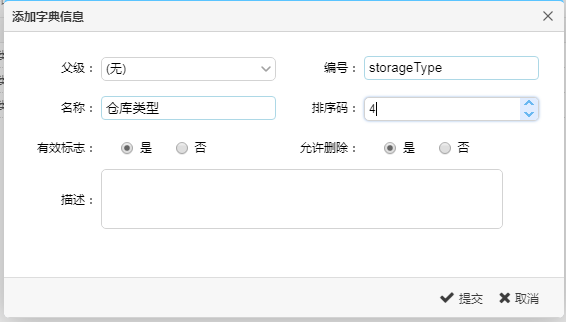


图 4-19 添加字典信息页面

添加成功后，在“字典管理”中选中“仓库类型”，页面如图 4-20所示：



图 4-20 字典管理

点击“添加”按钮，弹出如下图 4-21所示弹出框，按照图中所示的信息添加一条数据。



图 4-21 添加字典值信息页面

以上操作完成后，在WebApp->BaseData->Storage->Index.aspx文件下的datagrid控件中，添加以下代码。storageType对应字典类型的编码。

|  |
| --- |
| $gridList.datagrid({  toolbar: '#gridStorage-tb',  url: basePath + 'Ashx/BaseData/StorageHandler.ashx?action=FindList',  columns: [[  { title: '主键', field: 'F\_Id', hidden: true },  { title: '编号', field: 'F\_Code', width: 100, align: 'center' },  { title: '名称', field: 'F\_Name', width: 120, align: 'center' },  { title: '部门', field: 'F\_OrganizeId', width: 120, align: 'center'},  { title: '库管员', field: 'F\_Manager', width: 120, align: 'center'},  { title: '联系方式', field: 'F\_Phone', width: 100, align: 'center' },  { title: '仓库高度(m)', field: 'F\_Height', width: 120, align: 'center' },  { title: '仓库面积(㎡)', field: 'F\_Area', width: 120, align: 'center' },  {  title: '仓库类型', field: 'F\_Type', width: 80, align: 'center',  formatter: function (value, row, index) {  return $top.ltcache.dictItems["storageType"][value];  }  },  {  title: '有效标志', field: 'F\_EnabledFlag', width: 100, align: 'center',  formatter: function (value, row, index) {  return value == 1 ? '<i class="fa fa-toggle-on"></i>' : '<i class="fa fa-toggle-off"></i>';  }  },  { title: '排序码', field: 'F\_SortNum', width: 60, align: 'center' }    ]],  onBeforeLoad: function (param) {  param.keyword = $('#keyword').textbox('getValue');  }    }); |

代码添加完成后，打开“仓库管理”页面，根据字典编号查出字典类型名称，如下图 4-22所示：

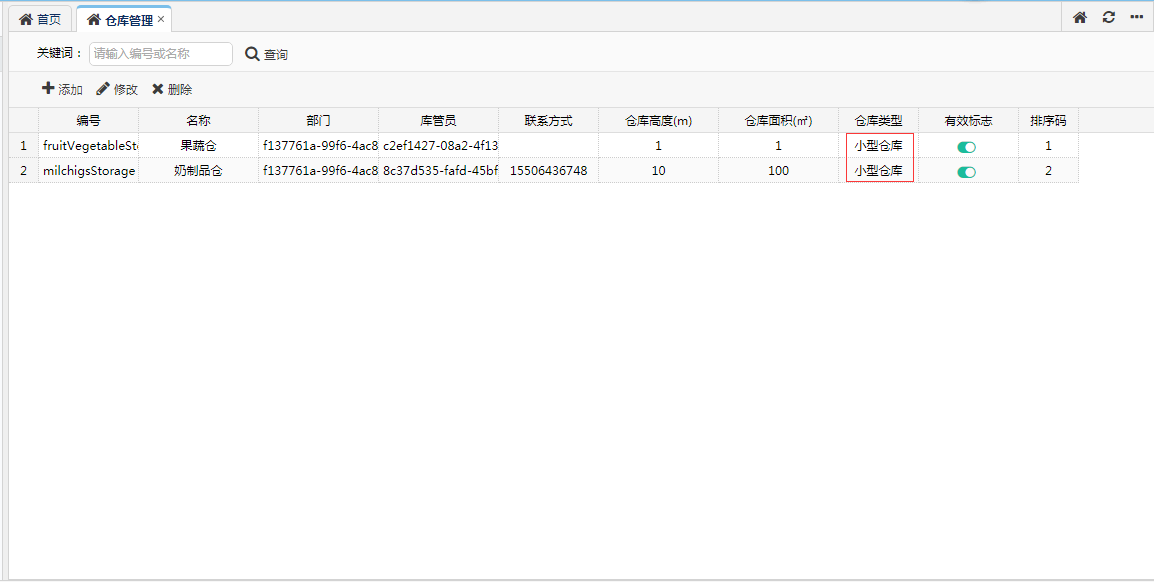


图 4-22 仓库管理页面

### 扩展字典选择字段

在BaseData文件夹下的Storage文件夹下的Index.aspx页面中，在<div class="queryPanel">标签内添加以下代码：

|  |
| --- |
| <div class="queryPanel">  <span>关键词：</span>  <input id="keyword" class="easyui-textbox" prompt="请输入编号或名称"/>  <span>仓库类型：</span>  <input id="fType"/>  <a id="btnQuery" class="easyui-linkbutton"><span class="fa fa-search"></span>查询</a>  </div> |

在jQuery中添加以下代码，combobox是下拉框控件(参照easyui用户手册使用)，storageType为字典类型的编码。

|  |
| --- |
| $('#fType').combobox({  url: basePath + 'Ashx/Common.ashx?action=GetDictItems&dictCode=storageType',  panelHeight: 80  }) |

storageType属于字典数据，在字典管理中可以添加。在系统管理功能中找到字典值管理，点击左侧的仓库类型，即可对该类型进行添加、修改、删除等操作。页面如图 4-23所示：

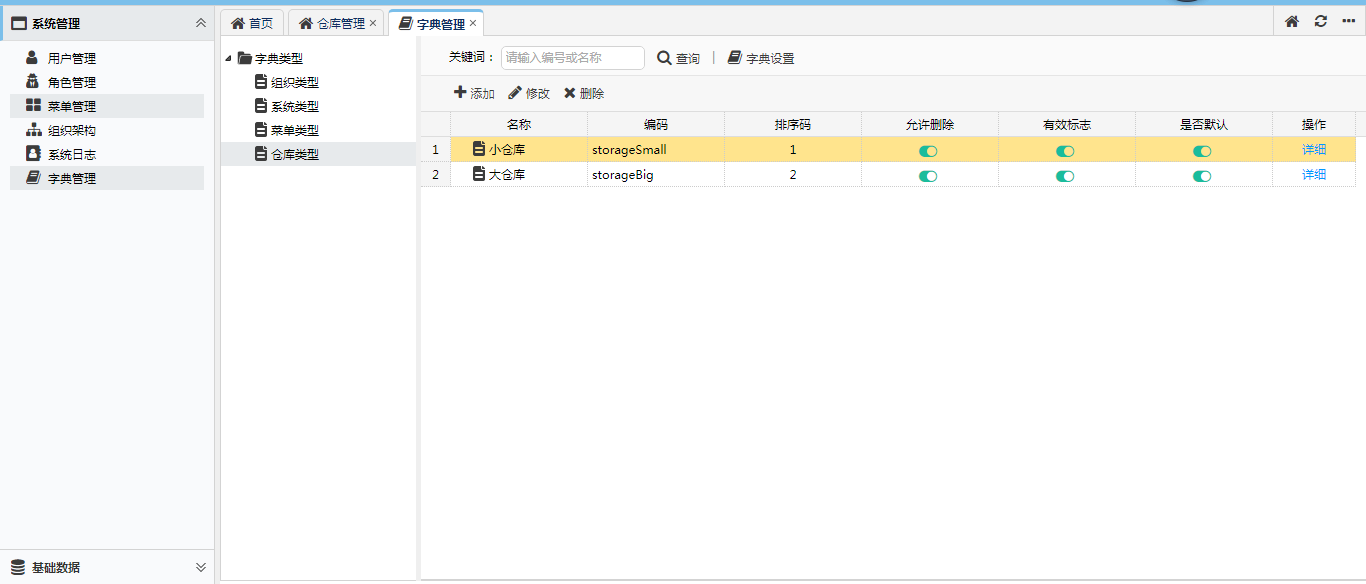


图 4-23 字典值管理页面

在datagrid的onBeforeLoad属性中添加查询条件，代码如下：

|  |
| --- |
| param.fType = $("#fType").combobox("getValue"); |

因为Handler层、Bll层都继承了封装好的接口，这里只需要在LT.Dal->BaseData->StorageDal文件中的FindList方法中添加以下代码：

|  |
| --- |
| public override List<StorageModel> FindList(Dictionary<string, string> queryParam)  {  string sql = @"select storage.\* from Bas\_Storage storage where F\_DeleteFlag=0";  List<DbParameter> paramters = new List<DbParameter>();  if (queryParam.ContainsKey("keyword"))  {  sql += "and (storage.F\_Name like @F\_Name or storage.F\_Code like @F\_Code)";  paramters.Add(new SqlParameter("@F\_Code", SqlDbType.NVarChar) { Value = string.Format("%{0}%", queryParam["keyword"]) });  paramters.Add(new SqlParameter("@F\_Name", SqlDbType.NVarChar) { Value = string.Format("%{0}%", queryParam["keyword"]) });  }  if (queryParam.ContainsKey("fType") && !string.IsNullOrEmpty(queryParam["fType"]))  {  sql += "and storage.F\_Type=@F\_Type ";  paramters.Add(new SqlParameter("@F\_Type", SqlDbType.NVarChar) { Value = queryParam["fType"] });  }  sql += "order by F\_SortNum";  List<StorageModel> storages = new List<StorageModel>();  using (DbDataReader dr = DbHelper.ExecuteReader(sql, paramters.ToArray()))  {  while (dr.Read())  {  StorageModel storage = ReaderToModel<StorageModel>(dr);  storages.Add(storage);  }  }  return storages;  } |

以上代码添加完后，打开“仓库管理”页面如下图 4-24所示：

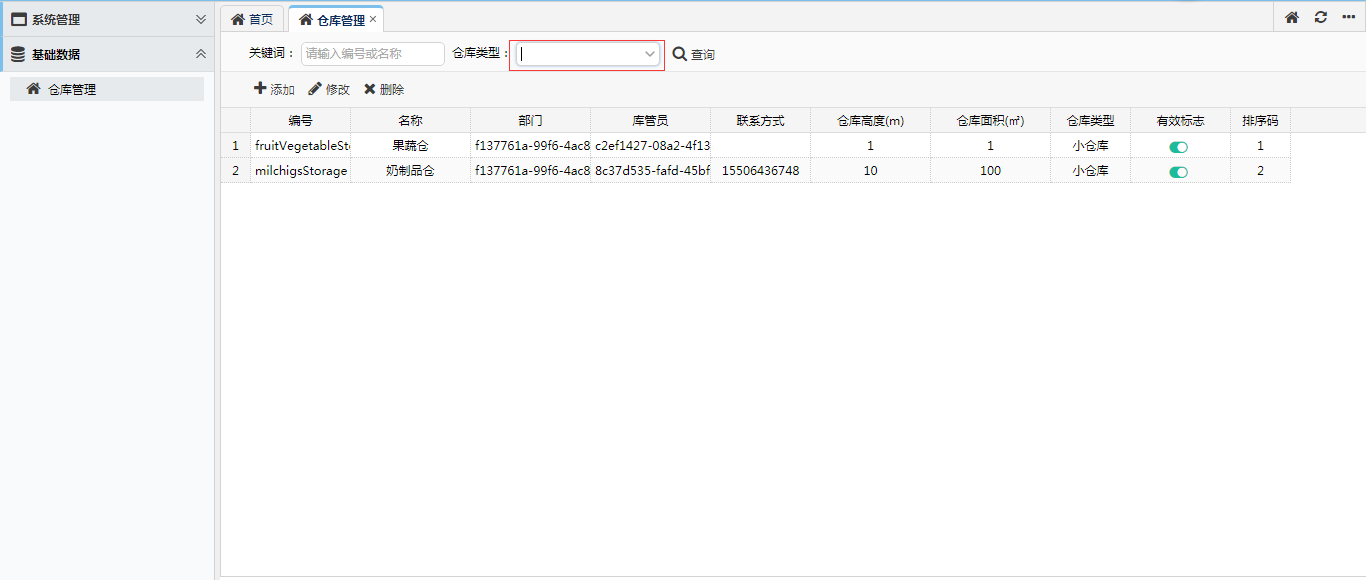


图 4-24 仓库管理页面

在“仓库类型”中选择“大仓库”，点击“查询”按钮，查询结果如下图 4-25所示。

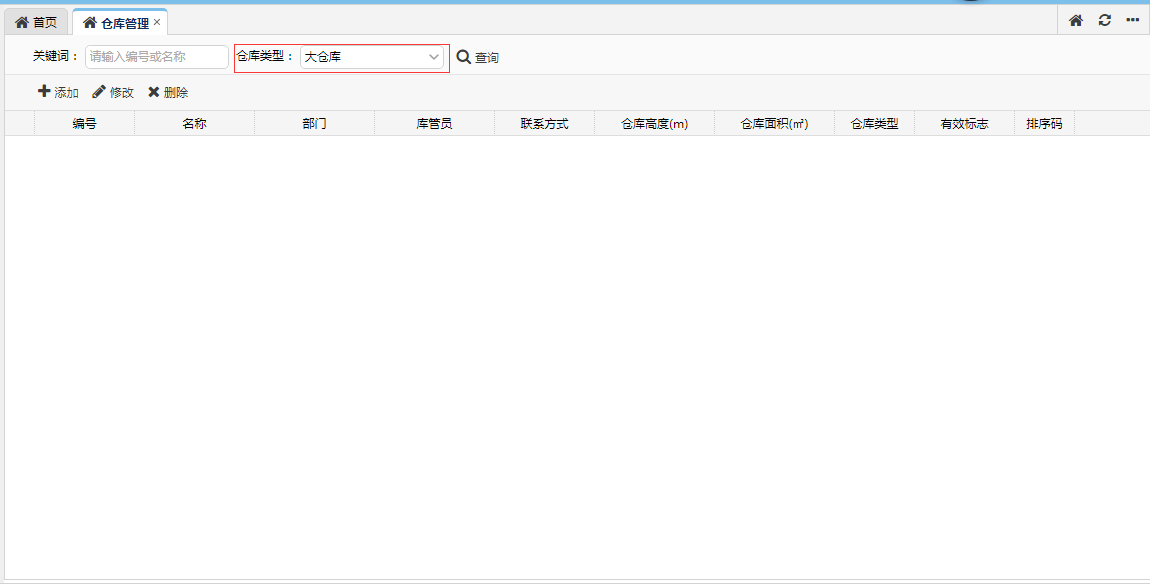


图 4-25 仓库管理页面

### 使用分页

将BaseData->Storage->Index.aspx文件的datagrid代码修改为以下代码：

|  |
| --- |
| $gridList.datagrid({  pagination: true,  toolbar: '#gridStorage-tb',  url: basePath + 'Ashx/BaseData/StorageHandler.ashx?action=FindPageList',  columns: [[  { title: '主键', field: 'F\_Id', hidden: true },  { title: '编号', field: 'F\_Code', width: 100, align: 'center' },  { title: '名称', field: 'F\_Name', width: 120, align: 'center' },  { title: '部门', field: 'F\_OrganizeId', width: 120, align: 'center'},  { title: '库管员', field: 'F\_Manager', width: 120, align: 'center'},  { title: '联系方式', field: 'F\_Phone', width: 100, align: 'center' },  { title: '仓库高度(m)', field: 'F\_Height', width: 120, align: 'center' },  { title: '仓库面积(㎡)', field: 'F\_Area', width: 120, align: 'center' },  {  title: '仓库类型', field: 'F\_Type', width: 80, align: 'center',  formatter: function (value, row, index) {  return $top.ltcache.dictItems["storageType"][value];  }  },  {  title: '有效标志', field: 'F\_EnabledFlag', width: 100, align: 'center',  formatter: function (value, row, index) {  return value == 1 ? '<i class="fa fa-toggle-on"></i>' : '<i class="fa fa-toggle-off"></i>';  }  },  { title: '排序码', field: 'F\_SortNum', width: 60, align: 'center' }    ]],  onBeforeLoad: function (param) {  param.keyword = $('#keyword').textbox('getValue');  param.fType = $("#fType").combobox("getValue");  }  }); |

后台在LT.Dal->BaseBata->Storage.aspx的FindPageList方法下加入以下代码：

|  |
| --- |
| string sql = @"select storage.\* from Bas\_Storage storage where F\_DeleteFlag=0";  List<DbParameter> paramters = new List<DbParameter>();  if (queryParam.ContainsKey("keyword"))  {  sql += "and (storage.F\_Name like @F\_Name or storage.F\_Code like @F\_Code)";  paramters.Add(new SqlParameter("@F\_Code", SqlDbType.NVarChar) { Value = string.Format("%{0}%", queryParam["keyword"]) });  paramters.Add(new SqlParameter("@F\_Name", SqlDbType.NVarChar) { Value = string.Format("%{0}%", queryParam["keyword"]) });  }  if (queryParam["fType"]!="")  {  sql += "and storage.F\_Type=@F\_Type ";  paramters.Add(new SqlParameter("@F\_Type", SqlDbType.NVarChar) { Value = queryParam["fType"] });  }  string pageSql = GetPageSql(sql, pagination, "F\_SortNum", paramters.ToArray());  List<StorageModel> storages = new List<StorageModel>();  using (DbDataReader dr = DbHelper.ExecuteReader(pageSql, paramters.ToArray()))  {  while (dr.Read())  {  StorageModel storage = ReaderToModel<StorageModel>(dr);  storages.Add(storage);  }  }  return storages; |

页面下方显示当前页数和总页数，如下图 4-26所示：

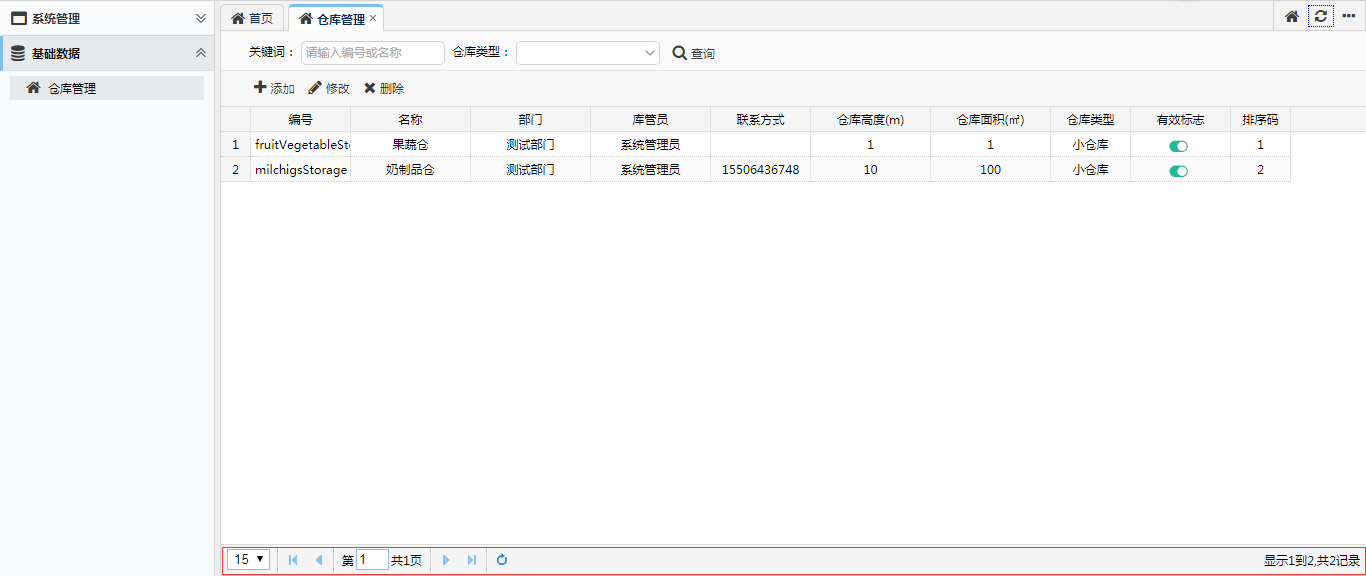


图 4-26 仓库管理页面

### 加入按钮权限

在按钮中添加btnauth="true"属性，代码如下所示：

|  |
| --- |
| <a id="btnAdd" btnauth="true" class="easyui-linkbutton"><span class="fa fa-plus"></span>添加</a>  <a id="btnEdit" btnauth="true" class="easyui-linkbutton"><span class="fa fa-pencil"></span>修改</a>  <a id="btnDelete" btnauth="true" class="easyui-linkbutton"><span class="fa fa-remove"></span>删除</a> |

在BaseData->Storage->Index.aspx文件下添加控制按钮权限的代码，如下所示：

|  |
| --- |
| var funcCode = getActiveFuncCode();  checkBtnAuthsByCode(funcCode); |

刷新浏览器，点开仓库管理页面，添加、修改、删除按钮因为没有权限而不显示，如图 4-27所示：

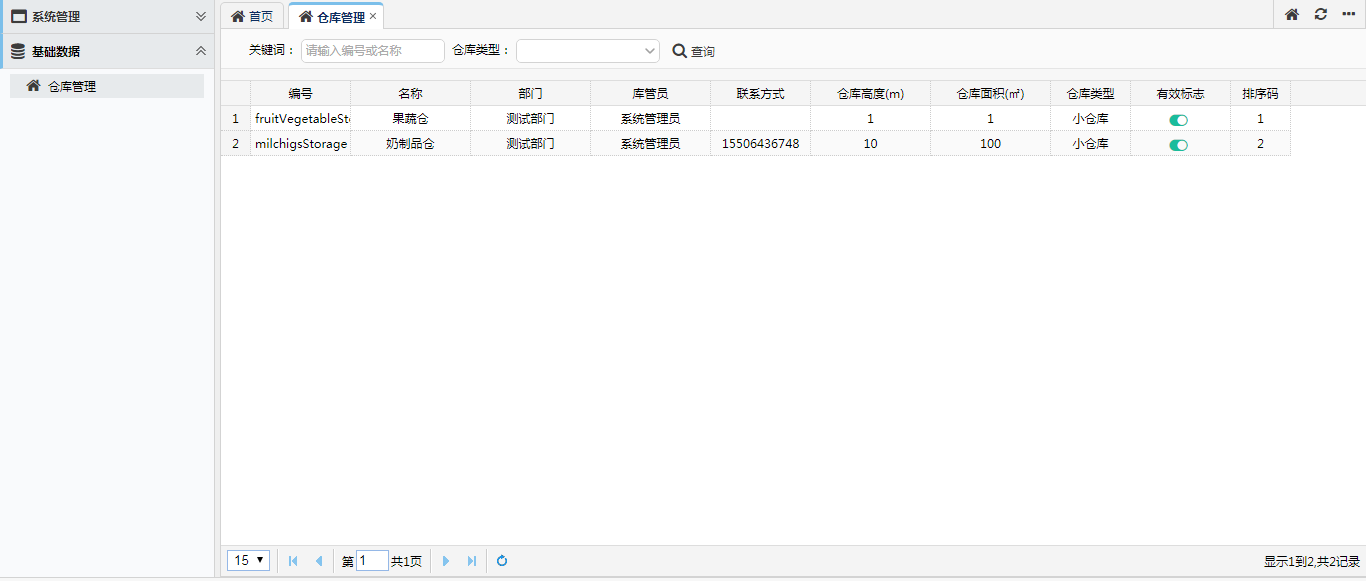


图 4-27 仓库管理页面

在“系统管理”中找到“菜单管理”，选中“基础数据”模块下的“仓库管理”，点击“按钮管理”弹出菜单按钮管理页面如下图 4-28所示：



图 4-28 菜单按钮管理

点击“添加”按钮弹出添加按钮信息页面，页面如下图 4-29所示，分别添加“添加”、“修改”、“删除”三个按钮的信息。添加按钮信息页面的编号对应着按钮的id。

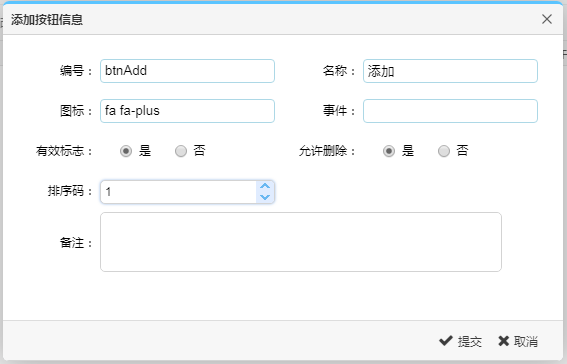


图 4-29 添加按钮信息页面

当增加、删除、修改按钮都添加完之后，在系统管理中的角色管理页面选择“testRole”这条数据，点击“赋予权限”按钮。选中下面区域的选项，点击提交即可。页面入下图 4-30所示：

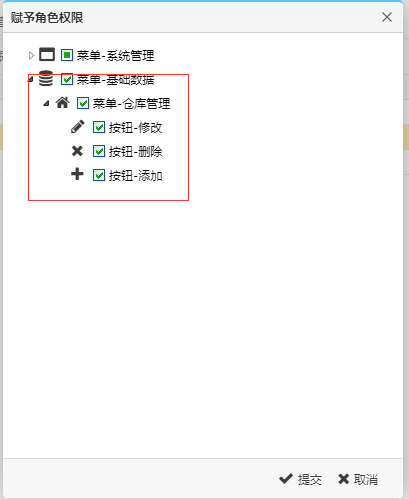


图 4-30 赋予角色权限页面

以上操作成功后刷新浏览器，点击仓库管理。添加、修改、删除按钮显示出来，如图 4-31所示：

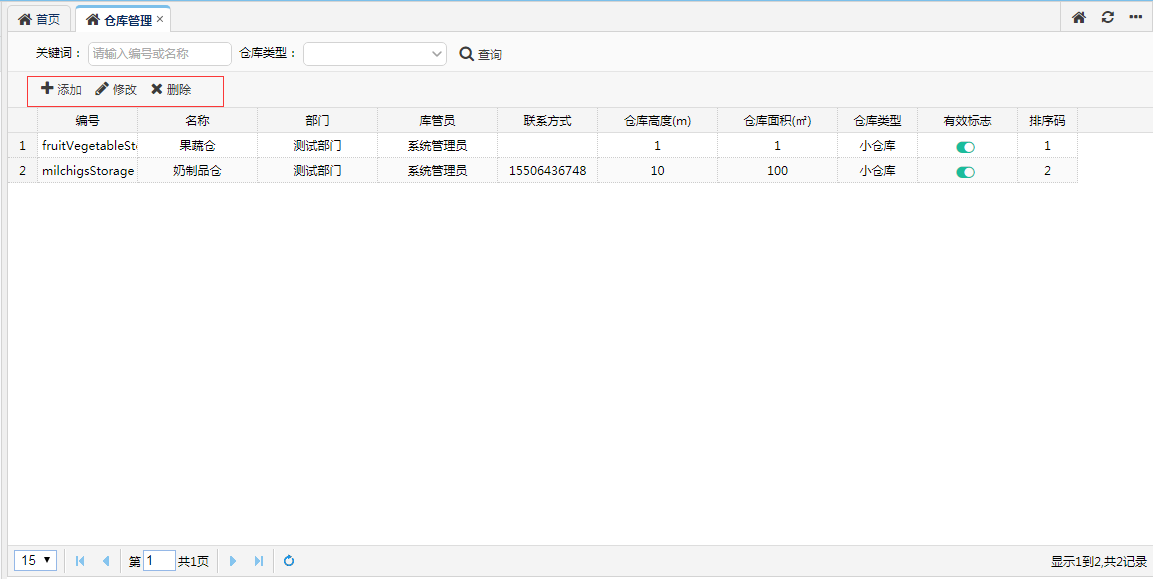


图 4-31 仓库管理页面

### 添加操作日志

在Ashx->BaseData->StorageHandler.ashx文件中，在StorageHandler类上面添加以下代码。因为这里继承了FormAshx<StorageModel>接口，所以只需要添加这一句代码即可。

|  |
| --- |
| [FuncMsg("基础数据", "仓库管理")] |

在仓库管理中修改一条数据，在系统日志中显示的数据入下图 4-32所示：

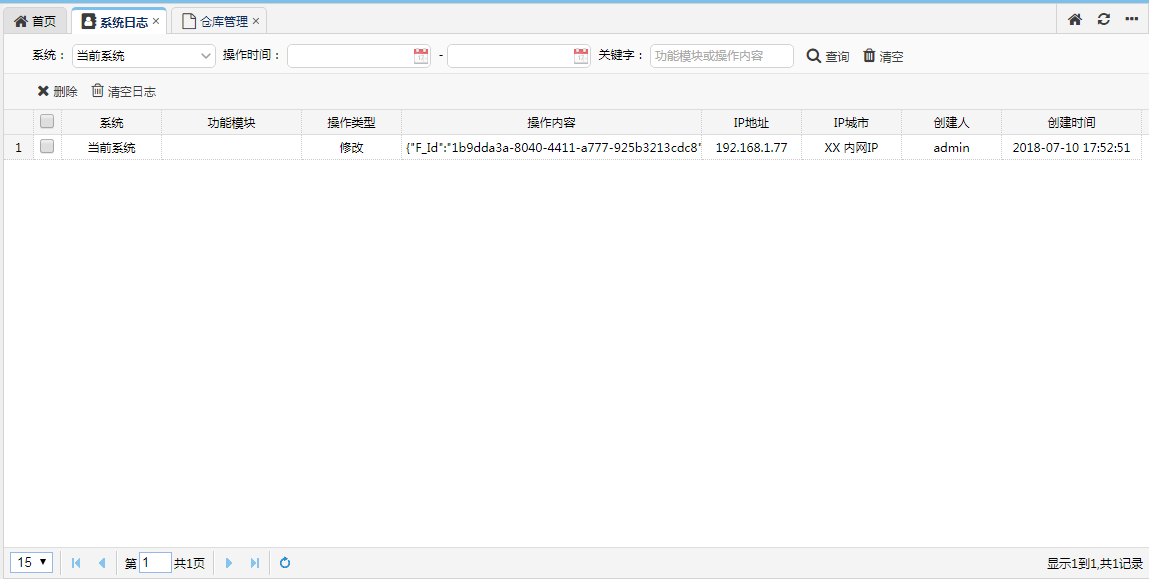


图 4-32 系统日志查询页面

### 说明树形结构

#### 树结构的选择框

以“系统管理”->“用户管理”的Form表单的“部门信息”下拉框为例。前台代码如下所示，调用“FindSelectTree”这个方法即可。

|  |
| --- |
| $('#F\_DepartmentId').combobox({  url: basePath + 'Ashx/SysManager/OrganizeHandler.ashx?action=FindSelectTree&fType=2',  valueField: 'id',  width: 175,  panelHeight: 80  }); |

页面如图 4-33所示：

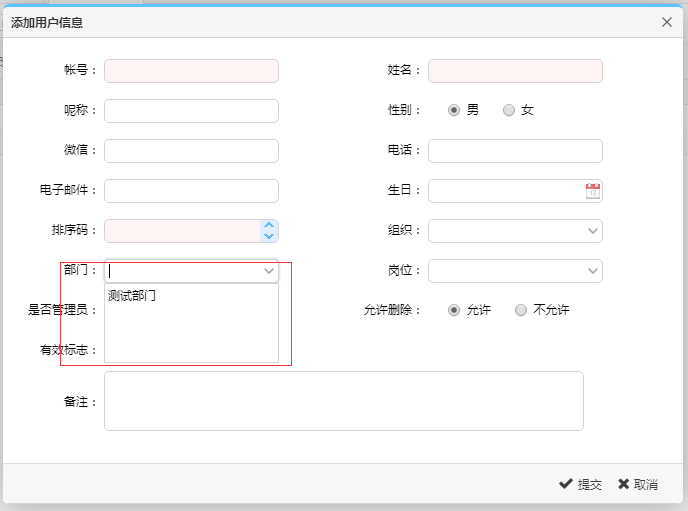


图 4-33 添加用户信息页面

#### 树结构查询

与datagrid不同的是，树结构的查询采用的是easyui的treegrid控件，查询表格的请求的方法由FindList改为FindTree。树结构可以参照组织架构(Organize)功能。

前端代码可参照WebApp->SysManager->Organize->Index.aspx文件。代码如下：

|  |
| --- |
| $gridList.treegrid({  striped: true,  idField: 'F\_Id', treeField: 'F\_Name',  fitColumns: true, toolbar: '#gridOrganize-tb',  url: basePath + 'Ashx/SysManager/OrganizeHandler.ashx?action=FindGridTree',  columns: [[  { title: '父级', field: 'F\_ParentId', width: 55, hidden: true },  { title: '名称', field: 'F\_Name', width: 120 },  { title: '编号', field: 'F\_Code', width: 100 },  { title: '简称', field: 'F\_ShortName', width: 120 },  {  title: '类型', field: 'F\_Type', width: 80,  formatter: function (value, row, index) {  return $top.ltcache.dictItems["organizeType"][value];  }  },  { title: '负责人', field: 'F\_ManagerId', width: 120 },  { title: '电话', field: 'F\_Phone', width: 120 },  { title: '联系地址', field: 'F\_Address', width: 120 },  { title: '排序码', field: 'F\_SortNum', width: 60, align: 'center' },  {  title: '允许删除', field: 'F\_EnableDelete', width: 70, align: 'center',  formatter: function (value, row, index) {  return value == 1 ? '<i class="fa fa-toggle-on"></i>' : '<i class="fa fa-toggle-off"></i>';  }  }, {  title: '有效标志', field: 'F\_EnabledFlag', width: 70, align: 'center',  formatter: function (value, row, index) {  return value == 1 ? '<i class="fa fa-toggle-on"></i>' : '<i class="fa fa-toggle-off"></i>';  }  },  {  title: '操作', field: 'cz', width: 60, align: 'center',  formatter: function (value, row, index) {  return '<a href="javascript:detail(\'' + row.F\_Id + '\')" class="row-operation">详细</a>';  }  }  ]],  onBeforeLoad: function (row, param) {  return getQueryParam(param);  },  onDblClickRow: function (row) {  detail(row.F\_Id);  }  }); |

组织架构后台与仓库管理所不同的是，如下表 4-2 差异所示：

表 4-2 差异

|  |  |  |
| --- | --- | --- |
|  | **treegrid** | **datagrid** |
| Handler层 | 继承TreeAshx<OrganizeModel> | 继承FormBll<StorageModel> |
| Bll层 | 继承TreeBll<OrganizeModel> | 继承FormBll<StorageModel> |
| Dal层 | 继承TreeDal<OrganizeModel> | 继承<StorageModel> |
| 查询请求调用 | FindTree | FindList和FindPageList |

具体的实现请参照织架构(Organize)的代码逻辑。

Handler层：Ashx->SysManager->OrganizeHandler.ashx文件。代码如下：

|  |
| --- |
| public class OrganizeHandler : TreeAshx<OrganizeModel>  {  private readonly OrganizeBll bll = new OrganizeBll();  /// <summary>  /// 框架中定义的必须要实现的方法，获取逻辑处理层的对象。  /// </summary>  /// <returns></returns>  public override TreeBll<OrganizeModel> GetBll()  {  return bll;  }  /// <summary>  /// 获取查询的参数信息，并转换到dictionary中，该对象直接传输到DAL层。  /// </summary>  /// <param name="context"></param>  /// <returns></returns>  protected override Dictionary<string, string> GetQueryParam(HttpContext context)  {  Dictionary<string, string> queryParams = base.GetQueryParam(context);  return queryParams;  }  } |

Bll层：LT.Bll->BaseData->StorageBll.cs文件。代码如下：

|  |
| --- |
| public class OrganizeBll : TreeBll<OrganizeModel>  {  OrganizeDal dal = new OrganizeDal();  /// <summary>  /// 获取数据处理层的对象  /// </summary>  /// <returns></returns>  public override TreeDal<OrganizeModel> GetDal()  {  return dal;  }  /// <summary>  /// 对树查询的内容进行筛选操作  /// </summary>  /// <param name="modelList"></param>  /// <param name="queryParam"></param>  /// <returns></returns>  protected override List<OrganizeModel> FilterGridTree(List<OrganizeModel> modelList, Dictionary<string, string> queryParam)  {  var con = ExtLinq.True<OrganizeModel>();  if (queryParam.ContainsKey("keyword"))  {  con = con.And(t => t.F\_Name.Contains(queryParam["keyword"]));  con = con.Or(t => t.F\_ShortName != null && t.F\_ShortName.Contains(queryParam["keyword"]));  }  modelList = modelList.TreeWhere(con.ToPredicate());  modelList = modelList.OrderBy(t => t.F\_SortNum).ToList(); // 使用TreeWhere后会打乱原排序，这里进行重新排序  return modelList;  }  } |

Dal层：LT.Dal->BaseData->StorageDal.cs文件。代码如下：

|  |
| --- |
| public class OrganizeDal : TreeDal<OrganizeModel>  {  public override string TableName  {  get { return "Sys\_Organize"; }  }  /// <summary>  /// 获取树结构的所有信息  /// </summary>  /// <param name="queryParam">查询的参数信息</param>  /// <returns></returns>  public override List<OrganizeModel> FindTree(Dictionary<string, string> queryParam)  {  string sql = @"select \* from Sys\_Organize where F\_DeleteFlag=0 ";  DbParameter[] parameters = null;  if (queryParam.ContainsKey("fType"))  {  sql += "and F\_Type=@F\_Type";  parameters = new DbParameter[]{  new SqlParameter("@F\_Type", SqlDbType.VarChar){ Value = queryParam["fType"] }  };  }  sql += " order by F\_SortNum";  List<OrganizeModel> organizes = new List<OrganizeModel>();  using (DbDataReader dr = DbHelper.ExecuteReader(sql, parameters))  {  while (dr.Read())  {  OrganizeModel organize = ReaderToModel<OrganizeModel>(dr);  organizes.Add(organize);  }  }  return organizes;  }  } |

# 特定功能

## 新建实体类ID

|  |  |
| --- | --- |
| **继承接口** | **说明** |
| 后台Handler层继承FormAshx<TModle>或TreeAshx<TModel>接口 | 只要继承前面的接口，自动生成实体类id |
| 实体类(TModel)，继承 BaseEntity<TEntity>接口，并继承ICreateAuth接口的。 | 可以调用BaseEntity中的Create()、NewGuid()方法创建实体类id。  需要先实例化实体类，例如：  StorageModel model=new StorageModel();  model.Create();//创建“实体类id”、“创建人”、“创建时间”，给这三个字段赋值。  model.NewGuid();//创建实体类id; |
| 实体类（TModel），继承 BaseEntity<TEntity>接口，但不继承ICreateAuth接口的。 | 调用BaseEntity中的NewGuid()方法创建实体类id。 |

## 获取当前用户信息

前台获取：jQuery中调用 $top.ltcache.userMsg方法即可获取用户的姓名、用户账号、昵称等信息。

后台获取：调用Ashx->Master.ashx请求下的LoadAllCache方法，代码如下所示。蓝色部分即为获取的用户当前信息。

|  |
| --- |
| doAjax({  url: "Ashx/Master.ashx",  action: "LoadAllCache",  async: false,  success: function (e) {  $.extend(cache.userMsg, e.userMsg);  cache.dictItems = e.dictItems;  cache.menuAuths = $.parseJSON(e.menuAuths);  cache.btnAuths = e.btnAuths;  }  }); |

## 添加操作日志

继承TreeAshx<TModel>、FormAshx<TModel>这两个接口的只需要，在Handler层添加“[FuncMsg(模块, 功能)]”这段代码，例如 [FuncMsg("系统管理", "用户管理")].

如果继承的是BaseAshx接口，除了上面需要加的代码以外，还需要添加一些配置信息，例如“系统设置功能”，在Ashx->SysManager->ConfigsHandler.ashx下的SubmitForm方法，当修改完成的时候执行“ AddSysLog(操作类型, 操作内容)”方法。例如：

|  |
| --- |
| if (result)  {  //修改配置成功后，写入系统日志  string optContent = "修改后配置为：";  foreach (ConfigsModel configsModel in changeObj)  {  optContent += string.Format("{0}:{2}; ", configsModel.F\_Name, configsModel.F\_Id, configsModel.F\_Value);  }  AddSysLog("修改", optContent);  return SuccessResult("修改成功！");  } |

## 系统自定义配置

例如在Config->appSetting.config文件夹下添加一条默认数据库的配置信息代码，如下所示。

|  |
| --- |
| <add key="defaultDB" value="SqlServer"/> |

后台读取默认数据库的调用Configs.GetSetting("defaultDB")方法获取。defaultDB指的appSetting.config配置信息中的key属性。

## 字典使用说明

字典分为字典类型和字典值，字典类型相当于字典值的父类。如下图 5-1所示，左侧的为字典类型，右侧的为字典值。

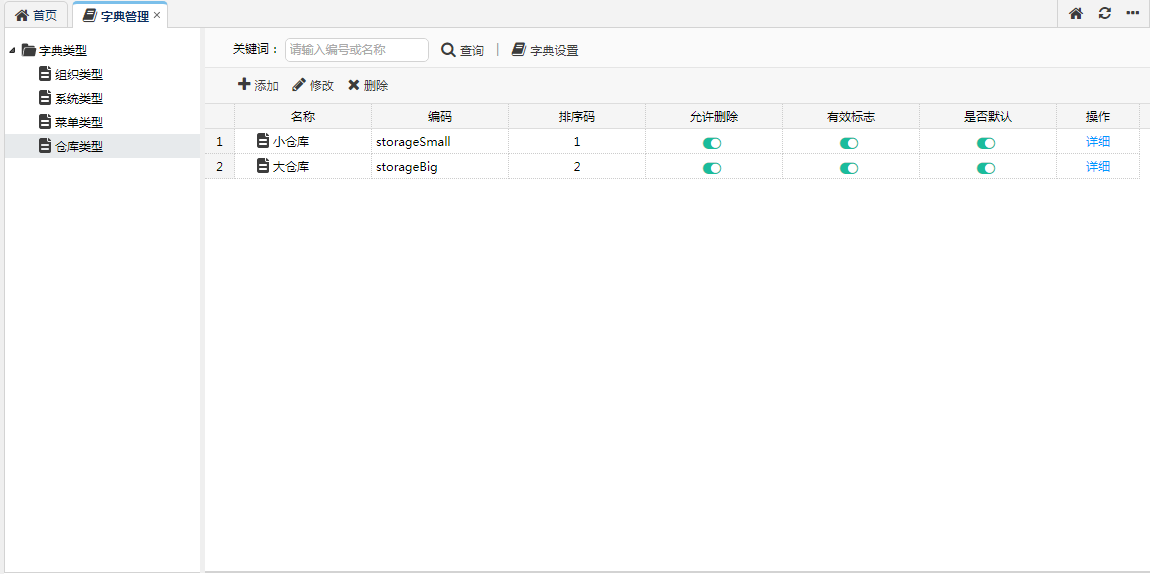


图 5-1 字典管理页面

字典类型可以通过点击“字典设置”按钮来添加如图 5-2所示：

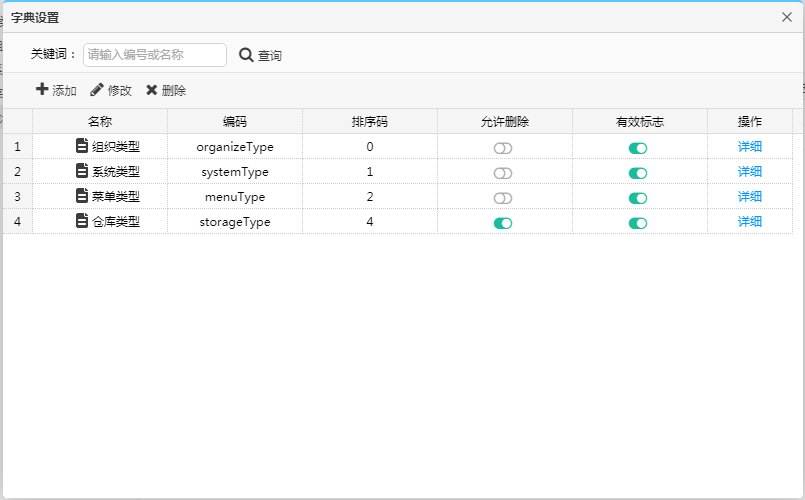


图 5-2 字典设置页面

点击添加按钮，添加字典类型的信息，弹出如图 5-3所示的窗口。

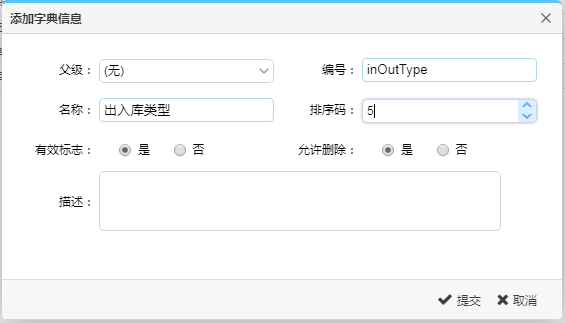


图 5-3 添加字典信息页面

添加完成后，在字典管理页面中增加一个字典类型，如下图 5-4所示：

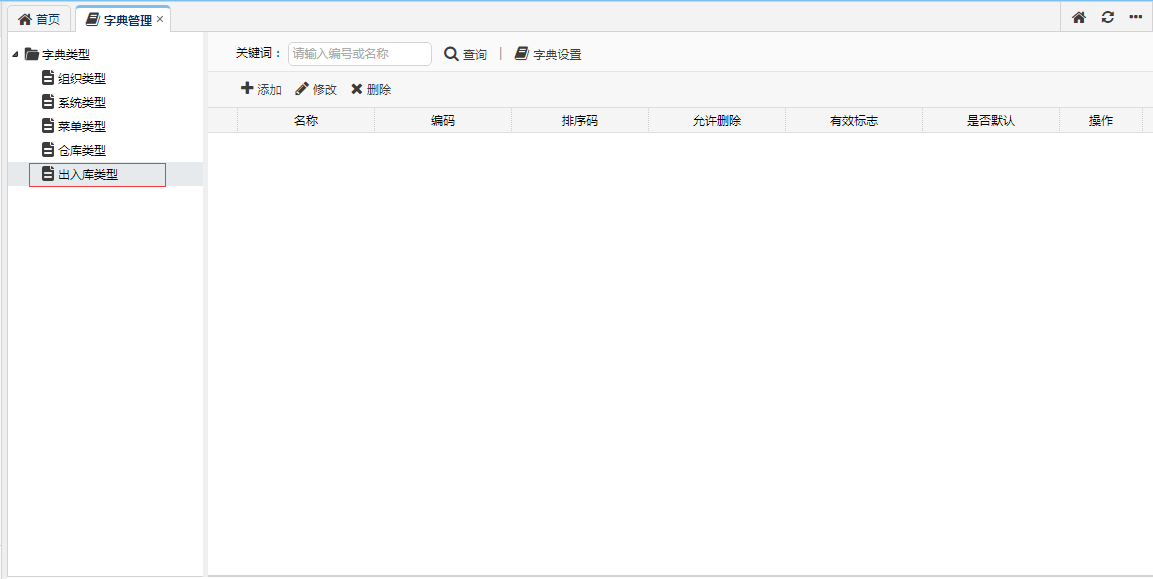


图 5-4 字典管理页面

添加“出入库类型”下的字典值信息。点击图 5-4中的“添加”按钮，弹出如下图 5-5所示的窗口，填写相关信息。

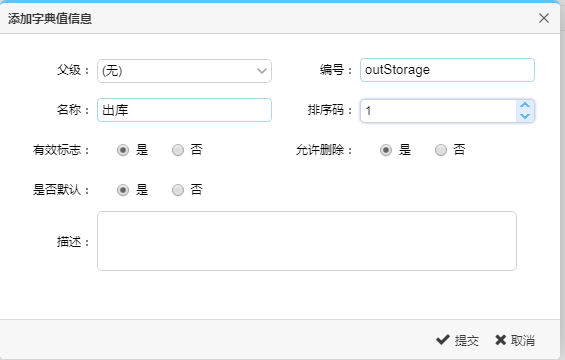


图 5-5 添加字典值信息页面

添加完成后字典管理页面展示如图 5-6所示：

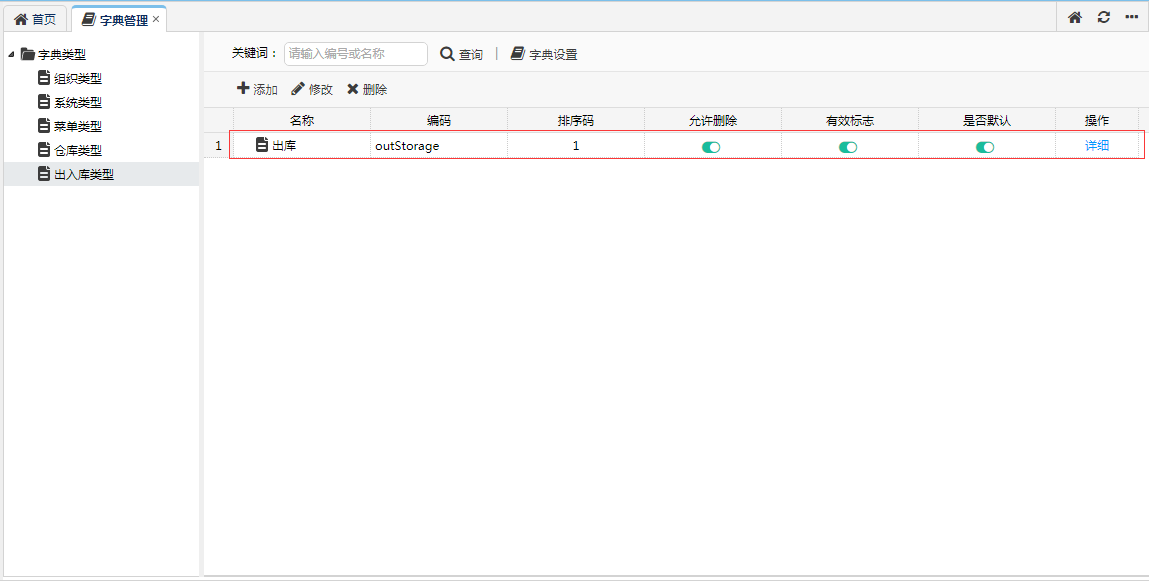


图 5-6 字典管理页面

以上是字典类型和字典值的关系以及添加操作。

字典使用说明以“仓库管理”中的“仓库类型”下拉框为例。找到WebApp->BaseData->->Storage->Index.aspx文件中的“仓库类型”下拉框，前端代码如下所示。

|  |
| --- |
| $('#fType').combobox({  url: basePath + 'Ashx/Common.ashx?action=GetDictItems&dictCode=storageType',  panelHeight: 80  }); |

“storageType”指的是字典类型的编码，通过这个请求获取这个字典编码下的字典值，这个编码对应的字典管理中的信息为下图 5-7所示。



图 5-7 仓库管理页面

对应仓库管理的仓库类型下拉框查询出的结果如下图 5-8所示：

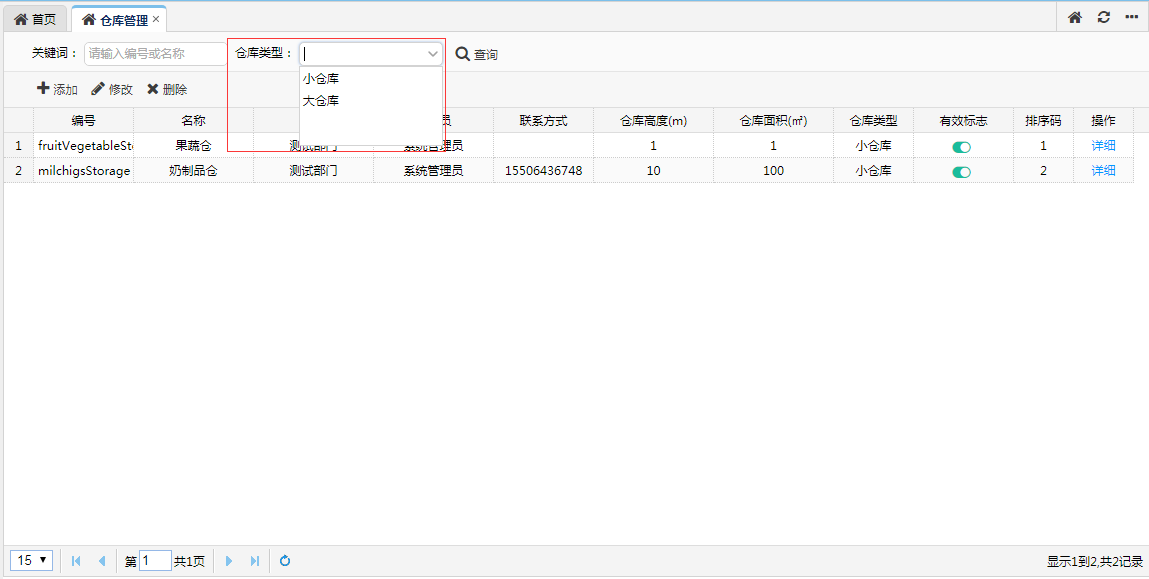


图 5-8 仓库管理页面

## 数据库切换

### SqlServer

### MySql

# 技术介绍

## easyUI

easyui是一种基于jQuery的用户界面插件集合。

easyui为创建现代化，互动，JavaScript应用程序，提供必要的功能。

使用easyui你不需要写很多代码，你只需要通过编写一些简单HTML标记，就可以定义用户界面。

easyui是个完美支持HTML5网页的完整框架。

easyui节省您网页开发的时间和规模。

## 前端集成插件

插件的说明如下表 6-1所示：

表 6-1 插件说明

|  |  |
| --- | --- |
| **插件** | **说明** |
| Easyui | 开发中一种便于前端开发的插件，路径：Resources->js->easyui文件夹下 |
| Highcharts | 是一种用来统计的插件，路径Resources->js->highcharts文件夹下。 |
| Flexslider | 轮播的插件（未用到），路径Resources->js->flexslider文件夹下。 |
| Jplayer | 媒体播放器（未用到），路径Resources->js->jplayer文件夹下。 |
| Jqprint | 打印的插件（未用到），路径Resources->js->jqprint文件夹下 |
| Layer | 弹出层插件（未用到），路径Resources->js->layer文件夹下。 |
| Ueditor | 富文本编辑器（未用到），路径Resources->js->ueditor文件夹下。 |

## 树结构层级码

树结构是基于easyui的treegrid控件，当需要去查询、删除一个树的所有根节点的时候可以用到层级码来查找。

层级码的编码规则为：#根节点id#|#父节点id#|#id#。

例如：系统管理的层级码为：

|  |
| --- |
| #76447f4c-6449-43b8-a5e5-225f4141bff0# --系统管理的Id |

系统管理->菜单管理，层级码格式为：

|  |
| --- |
| #76447f4c-6449-43b8-a5e5-225f4141bff0#|#76447f4c-6449-43b8-a5e5-225f4141bff2# --菜单管理的Id |

系统管理->菜单管理->添加按钮，层级码格式为：

|  |
| --- |
| #76447f4c-6449-43b8-a5e5-225f4141bff0#|#76447f4c-6449-43b8-a5e5-225f4141bff2#|#9396f851-b809-493e-8f5c-6856032bf6e6# --添加按钮的id |

# 系统类库说明

## LT.CoreMini

### 类库说明

主要是三层结构(Handler、Bll、Dal)的一些封装，还有公共方法的封装，还包括自定义的的异常、授权认证、登录用户的缓存信息、系统日志等。

### 业务相关功能

#### Bll

BaseBll是基础的逻辑处理的类继承IBaseBll接口，IBaseBll接口目前没有写方法后面可能会加入方法。FormBll和TreeBll都继承BaseBll。

TreeBll<TModel>是基于treegrid的查询，FormBll<TModel>是基于datagrid控件的查询。FormBll<TModel>和TreeBll<TModel>对简单的增删改查的操作进行了封装。方法说明如下表 7-1 方法说明所示：

表 7-1 方法说明

|  |  |  |  |
| --- | --- | --- | --- |
| **FormBll<TModel>** | **说明** | **TreeBll<TModel>** | **说明** |
| GetDal() | 获取数据处理层的对象，虚方法，需要在后台进行实现 | GetDal() | 数据处理层的对象获取，虚类，需要被继承才可以使用。 |
| FindForm(string fId) | 根据表单的ID获取当前表单的信息 | FindForm(string fId) | 根据表单的ID获取当前表单的信息 |
| AddForm(TModel model) | 表单信息添加 | AddForm(TModel model) | 表单信息添加 |
| EditForm(TModel model) | 表单信息修改 | EditForm(TModel model) | 表单信息修改 |
| DeleteForm(string fId) | 表单信息删除 | DeleteForm(string fId) | 表单信息删除 |
| FindList(Dictionary<string, string> queryParam) | 获取列表数据 | FindTree(Dictionary<string, string> queryParam) | 查询树的所有数据，并进行筛选后，返回到Action层 |
| FindPageList(Dictionary<string, string> queryParam, Pagination pagination) | 带有分页的列表信息查询 | FindTree(Dictionary<string, string> queryParam) | 查询树列表时，对树的数据进行筛选 |
|  |  | GetModelLayers(string parentId, string fId) | 获取当前对象的层级信息 |
|  |  | IsExistChild(string fId) | 检测当前节点是否存在子节点 |

#### DAL

DbFactory是调度数据库的类，包含一些数据库的配置，方法说明如下表 7-2：

表 7-2 DbFactory类说明

|  |  |
| --- | --- |
| **方法** | **说明** |
| GetDbHelper() | 构造函数，主要是更新日志的配置文件 |
| GetDbHelper(string connKey) | 获取数据库的操作对象，根据配置信息 |
| RemoveDbHelperTemp(string connKey) | 释放并删除掉一个数据库连接的信息 |
| GetConnSetting(string configName) | 得到web.config里配置项的数据库连接信息。 |

IDbHelper是数据库操作基础类的逻辑处理方法。

MSSQLHelper是SqlServer数据库操作的类方法说明如下表 7-3所示

表 7-3 MSSQLHelper类的说明

|  |  |
| --- | --- |
| **方法** | **说明** |
| GetNewInstance(string con) | 非单例模式获取数据库帮助类 |
| GetInstance(string con) | 获取单例的信息 |
| TableExists(string tableName) | 表是否存在 |
| ColumnExists(string tableName, string columnName) | 判断是否存在某表的某个字段 |
| ExecuteNonQuery(string sqlString) | 执行SQL语句，返回影响的记录数 |
| ExecuteNonQuery(DbConnection connection, string sqlString) |
| ExecuteNonQuery(string sqlString, params DbParameter[] cmdParms) |
| ExecuteNonQuery(DbConnection connection, string sqlString, params DbParameter[] cmdParms) |
| ExecuteNonQuery(string sqlString, CommandType cmdType, params DbParameter[] cmdParms) |
| ExecuteScalar(string sqlString) | 执行一条计算查询结果语句，返回查询结果（object）。 |
| ExecuteScalar(DbConnection connection, string sqlString) |
| ExecuteScalar(string sqlString, params DbParameter[] cmdParms) |
| ExecuteScalar(DbConnection connection, string sqlString, params DbParameter[] cmdParms) |
| ExecuteScalar(string sqlString, CommandType cmdType, params DbParameter[] cmdParms) |
| ExecuteReader(string sqlString) | 执行查询语句，返回SqlDataReader ( 注意：调用该方法后，一定要对SqlDataReader进行Close ) |
| ExecuteReader(DbConnection connection, string sqlString) |
| ExecuteReader(string sqlString, params DbParameter[] cmdParms) |
| ExecuteReader(DbConnection connection, string sqlString, params DbParameter[] cmdParms) |
| ExecuteReader(string sqlString, CommandType cmdType, params DbParameter[] cmdParms) |
| ExecuteDataSet(string sqlString) | 执行查询语句，返回DataSet |
| ExecuteDataSet(DbConnection connection, string sqlString) |
| ExecuteDataSet(string sqlString, params DbParameter[] cmdParms) |
| ExecuteDataSet(DbConnection connection, string sqlString, params DbParameter[] cmdParms) |
| ExecuteDataSet(string sqlString, CommandType cmdType, params DbParameter[] cmdParms) |
| ExecuteSqls(List<CommandType> cmdTypes, List<string> cmdTexts, List<DbParameter[]> parameterses) | 进行事务处理，返回总共受影响的行数 |
| ExecuteSqls(DbConnection conn, List<CommandType> cmdTypes, List<string> cmdTexts, List<DbParameter[]> parameterses) |
| ExecuteSqls(List<string> sqlStrings, List<DbParameter[]> cmdParms) |
| ExecuteSqls(List<string> sqlStrings) |
| BackupOrRestore(string sqlString) | 执行SQL语句，返回是否执行成功,对于数据库备份和还原操作时使用 |

MySqlHelper是MySQL数据库操作的类方法说明如下表 7-4所示

表 7-4 MySqlHelper类的说明

|  |  |
| --- | --- |
| **方法** | **说明** |
| GetNewInstance(string con) | 非单例模式获取数据库帮助类 |
| GetInstance(string con) | 获取单例的信息 |
| TableExists(string tableName) | 表是否存在 |
| ColumnExists(string tableName, string columnName) | 判断是否存在某表的某个字段 |
| ExecuteNonQuery(string sqlString) | 执行SQL语句，返回影响的记录数 |
| ExecuteNonQuery(DbConnection connection, string sqlString) |
| ExecuteNonQuery(string sqlString, params DbParameter[] cmdParms) |
| ExecuteNonQuery(DbConnection connection, string sqlString, params DbParameter[] cmdParms) |
| ExecuteNonQuery(string sqlString, CommandType cmdType, params DbParameter[] cmdParms) |
| ExecuteScalar(string sqlString) | 执行一条计算查询结果语句，返回查询结果（object）。 |
| ExecuteScalar(DbConnection connection, string sqlString) |
| ExecuteScalar(string sqlString, params DbParameter[] cmdParms) |
| ExecuteScalar(DbConnection connection, string sqlString, params DbParameter[] cmdParms) |
| ExecuteScalar(string sqlString, CommandType cmdType, params DbParameter[] cmdParms) |
| ExecuteReader(string sqlString) | 执行查询语句，返回SqlDataReader ( 注意：调用该方法后，一定要对SqlDataReader进行Close ) |
| ExecuteReader(DbConnection connection, string sqlString) |
| ExecuteReader(string sqlString, params DbParameter[] cmdParms) |
| ExecuteReader(DbConnection connection, string sqlString, params DbParameter[] cmdParms) |
| ExecuteReader(string sqlString, CommandType cmdType, params DbParameter[] cmdParms) |
| ExecuteDataSet(string sqlString) | 执行查询语句，返回DataSet |
| ExecuteDataSet(DbConnection connection, string sqlString) |
| ExecuteDataSet(string sqlString, params DbParameter[] cmdParms) |
| ExecuteDataSet(DbConnection connection, string sqlString, params DbParameter[] cmdParms) |
| ExecuteDataSet(string sqlString, CommandType cmdType, params DbParameter[] cmdParms) |
| ExecuteSqls(List<CommandType> cmdTypes, List<string> cmdTexts, List<DbParameter[]> parameterses) | 使用现有的SQL事务执行一个sql命令（不返回数据集） |
| ExecuteSqls(DbConnection conn, List<CommandType> cmdTypes, List<string> cmdTexts, List<DbParameter[]> parameterses) |
| ExecuteSqls(List<string> sqlStrings, List<DbParameter[]> cmdParms) |
| ExecuteSqls(List<string> sqlStrings) |
| PrepareCommand(DbCommand cmd, DbConnection conn, DbTransaction trans, CommandType cmdType, string cmdText, DbParameter[] cmdParms) | 准备执行的command指令 |
| PrepareCommand(DbCommand cmd, DbConnection conn, string cmdText, DbParameter[] cmdParms) |
| PrepareCommand(DbCommand cmd, DbConnection conn, CommandType cmdType, string cmdText, DbParameter[] cmdParms) |
| BackupOrRestore(string sqlString) | 执行SQL语句，返回是否执行成功,对于数据库备份和还原操作时使用 |

IBaseDal是数据处理层的接口，BaseDal是IBaseDal的实现类，FormDal和TreeDal都继承BaseDal。

FormDal和TreeDal是基于对增删该查封装出来的数据处理类，方法说明如下表 7-5 方法说明所示：

表 7-5 方法说明

|  |  |  |  |
| --- | --- | --- | --- |
| **FormDal<TModel>** | **说明** | **TreeDal<TModel>** | **说明** |
| FindForm(string fId) | 根据表单的ID获取当前表单的信息 | FindForm(string fId) | 根据表单的ID获取当前表单的信息 |
| AddForm(TModel model) | 表单信息添加 | AddForm(TModel model) | 表单信息添加 |
| EditForm(TModel model) | 表单信息修改 | EditForm(TModel model) | 表单信息修改 |
| DeleteForm(string fId) | 表单信息删除 | DeleteForm(string fId) | 表单信息删除 |
| FindList(Dictionary<string, string> queryParam) | 获取列表数据 | FindTree(Dictionary<string, string> queryParam) | 查询树的所有数据，并进行筛选后，返回到Action层 |
| FindPageList(Dictionary<string, string> queryParam, Pagination pagination) | 带有分页的列表信息查询 | GetParentLayer(string parentId) | 获取父节点的层级信息 |
|  |  | ChangeAllLayer(string oldLevel, string newLevel) | 包括当前层级以及其所有子集的层级关系 |
|  |  | IsExistChild(string fId) | 检测当前节点是否存在子节点 |

#### Web

BaseAshx类是对于请求进行转换处理；定义前后台调用的规范；定义公共的方法，方法说明如下表 7-6所示：

表 7-6 BaseAshx类说明

|  |  |
| --- | --- |
| **方法** | **说明** |
| ProcessRequest(HttpContext context) | 默认访问方法，对所有方法进行调度 |
| CheckUserNoLogin(Type nowType, string action, HttpResponse response) | 判断用户未登录或者登录被挤掉，nowType">当前请求的处理类类型，action">当前亲你供求的方法，response">信息返回对象。 |

BaseAspxPage类是对于界面打开的验证以及权限等的验证进行了添加。只有受登录以及授权限制的功能集成该类即可，对于不需要该限制的，直接集成原生的Page类即可。

FormAshx类和TreeAshx类继承了BaseAshx。

FormAshx是单独表单处理的基类，这里采用虚类进行实现，无法直接进行调用，可以通过其继承类来使用这些实现的方法。

TreeAshx是对于树结构的表单项的处理方法， 区别于Form表单，这里的表单存在上下级关系，定义了树结构的数据可能会用到的一些方法，对于分页的树，这里不进行考虑。树分页后，实现的意义不大。

FormAshx和TreeAshx的方法说明如下表 7-7所示：

表 7-7 方法说明

|  |  |  |  |
| --- | --- | --- | --- |
| **FormAshx<TModel>** | **说明** | **TreeAshx<TModel>** | **说明** |
| GetBll() | 获取BLL层处理的操作类，这里为虚类，必须要在子类中进行实现的。 | GetBll() | 虚方法，获取逻辑处理层对象。 |
| FindForm(HttpContext context) | 根据表单的ID获取当前表单的信息 | FindForm(HttpContext context) | 根据表单的ID获取当前表单的信息 |
| SubmitForm(HttpContext context) | 表单信息添加、修改 | SubmitForm(HttpContext context) | 添加、修改表单数据 |
| FindSelectList(HttpContext context) | 获取下拉列表的项 | FindViewTree(HttpContext context) |  |
| DeleteForm(HttpContext context) | 表单信息删除 | DeleteForm(HttpContext context) | 表单信息删除 |
| FindList(HttpContext context) | 获取列表数据 | FindGridTree(HttpContext context) | 查询树列表 |
| FindPageList(HttpContext context) | 带有分页的列表信息查询 | FindSelectTree(HttpContext context) | 获取下拉列表的项，树结构的时候 |
| ModelListToSelectList(List<TModel> modelList) | 将查询的数据列表转化为下拉框选择用的对象 | FindSelectList(HttpContext context) | 获取下拉列表的项，非树结构时 |
|  |  | ModelListToSelectList(List<TModel> modelList) | 将查询的数据列表转化为下拉框选择用的对象 |

#### Entity

BaseEntity是实体对象的基类，方法说明如下表 7-8所示：

表 7-8 BaseEntity类说明

|  |  |
| --- | --- |
| **方法** | **说明** |
| Create() | 如果该实例继承了创建权限，则支持创建方法。否则会报异常。需要继承ICreateAuth接口。 |
| Modify(string fId) | 如果该实例继承了修改权限，则支持修改方法。否则会报异常。需要继承IModifyAuth接口。 |
| Remove() | 如果该实例继承了删除权限，则支持删除方法。否则会报异常。需要继承IDeleteAuth接口。 |
| NewGuid() | 获取全局唯一标识符guid |

BaseTreeEntity是树状结构的基类，继承BaseEntity类，方法说明如下表 7-9所示：

表 7-9 BaseTreeEntity类说明

|  |  |
| --- | --- |
| **方法** | **说明** |
| Create() | 如果该实例继承了创建权限，则支持创建方法。否则会报异常。 |
| Modify(string fId) | 如果该实例继承了修改权限，则支持修改方法。否则会报异常。 |
| Remove() | 如果该实例继承了删除权限，则支持删除方法。否则会报异常。 |

ICreateAuth是创建的权限，所有的实体类具有新建权限的都可以继承改类，用于确定该实体创建必须的字段。包含的参数有F\_Id、F\_CreateUserId、F\_CreateTime。

IDeleteAuth是删除的权限。（逻辑删除会用到，如果物理删除，直接删除即可，不需要定义这些字段），所有的实体类具有逻辑删除权限的都可以继承改类，用于确定该实体删除必须的字段。包含的参数有F\_Id、F\_DeleteFlag、F\_DeleteUserId、F\_DeleteTime。

IModifyAuth是修改的权限，所有的实体类具有修改权限的都可以继承改类，用于确定该实体修改必须的字段。包含的参数有F\_Id、F\_LastModifyUserId、F\_LastModifyTime。

### 公共资源

#### 公用实体

AjaxResult类是对ajax返回信息、结果的一个封装，还包含了表示 ajax 操作结果类型的枚举。

ComboSelectModel类是构造一个可以返回combobox格式信息的实体类。

JsonRpcResult类是JsonRPC协议格式的定义。

Pagination是分页信息辅助的实体类。

#### 自定义异常

CustomException类中自定义了几种异常类，方法如下表 7-10所示：

表 7-10 CustomException类的说明

|  |  |
| --- | --- |
| **方法** | **说明** |
| CustomException() | 自定义异常 |
| CustomException(string message) |
| CustomException(string msg, Exception innerException) |

## common.js

Common.js文件是对常用方法、easyUI默认值、easyUI方法扩展、jquery方法扩展、javascript原型扩展、通用方法、集成小类库的一个公共的js类库。

1. logger(msg)日志输方法，对前台console进行了封装。
2. alertSuccess(msg)提示成功信息的方法。
3. alertError(msg)提示错误信息的方法。
4. alertWarnning(msg)提示警告信息的方法。
5. confirm(msg, fn, title)提示信息的方法，带有确定取消，使用方式如下所示：

|  |
| --- |
| confirm("确定要删除选中数据？", function (e) {  if (e) {  方法、请求等。  };  }); |

1. openWindow(options)打开窗口的方法，打开一个窗口（不带有按钮），直接嵌入到主窗体中，可只写body部分的代码。

打开的窗口都是基于最顶层文档结构，输入值方便，返回值较为麻烦。

该方法，不建议使用，如果使用，对于窗口的ID避免系统中出现重复，并且获取数据以及关闭时，采用手动传值方式。使用方式参照一下代码：

|  |
| --- |
| openWindow({  id: 'AddFormUser',//是渲染窗体的标签的ID，建议使用win开头，尽量避免重复  title: '添加用户信息', //标题信息  url: 'WebApp/SysManager/User/Form.aspx',//窗口文件路径  wintype: 'add',//窗口状态，值有：normal-正常、add-添加、update-修改、view-查看  width: 680,//窗口的宽度  height: 500,//窗口的高度  data:data,//要传入窗口的值，在窗口中可以调用getWinData方法来获得  close: function (id, frameId) { //窗口关闭时的事件,事件参数包括窗口的id  $gridList.datagrid('reload');  }  }); |

1. openFullWindow(options),打开一个窗口（不带有按钮,全文档加载），完整的文档加载，存在iframe中。常用这个方法。

打开的窗口都是基于最顶层文档结构，输入值方便，返回值较为麻烦。

|  |
| --- |
| openFullWindow({  id: "EditFormUser",//是渲染窗体的标签的ID，建议使用win开头，尽量避免重复  title: '修改用户信息',//标题信息  url: 'WebApp/SysManager/User/Form.aspx?fId=' + selectItem.F\_Id,//窗口文件路径  wintype: 'edit',//窗口状态，值有：normal-正常、add-添加、update-修改、view-查看  width: 680,//窗口的宽度  height: 500,//窗口的高度  data:data,//要传入窗口的值，在窗口中可以调用getWinData方法来获得  close: function (id, frameId) {//窗口关闭时的事件,事件参数包括窗口的id  $gridList.datagrid('reload');  }  }); |

1. openBtnWindow(options), 加载带有按钮的窗体，按钮在窗体的下方。按钮只有两个，可后续自己扩展。

打开的窗口都是基于最顶层文档结构，输入返回值都较为方便，但是处理需要放在index页面。

1. getWinData(id)获取窗口的额外值，调用requestParam().winid获取窗口id,在调用getWinData(id)获取传过来的信息。
2. closeWindow(id)，关闭当前窗口，id为窗口id。
3. addTabs(title, url, param)添加一个选项卡面板。

title:选项卡的标题，也是选项卡的标识。

url：选项卡窗口对应的文件,即菜单中的连接。

param：data：要传入到选项卡的数据，可以通过getTabData方法来进行获取。

1. refreshTabs() 刷新当前选项卡面板，为了保留住面板传入的参数信息，这里只是针对iframe的内容进行更新即可。

使用方法：$('#allTab').tabs('refreshTabs')。

1. getTabData(title)，tab内页面调用获取传入的变量的方法。
2. getActiveFuncCode()通过tab的信息，获取当前功能的编码信息，即菜单管理中的编号。
3. checkBtnAuthsByCode(funcCode)，根据功能的编码，对于功能界面内的所有按钮进行权限的判定。funcCode 当前功能的编码，由getActiveFuncCode()方法获取，即菜单管理中的编号。
4. requestParam()获取url参数。
5. 去掉前导字符用trimStart()方法，str.trimStart(‘,’)。如果开头有“逗号”，将逗号去掉。
6. 去掉后导字符trimEnd()方法，str.trimEnd(‘,’)。如果结尾有“逗号”，将逗号去掉。
7. 去掉两端字符trim方法，str.trim(‘,’)。去掉两端的“逗号”。
8. window窗体默认属性：

|  |
| --- |
| if ($.fn.window) {  $.fn.window.defaults.resizable = false;//如果为true，允许列改变大小。  $.fn.window.defaults.collapsible = false;//定义是否显示可折叠按钮。  $.fn.window.defaults.minimizable = false;//定义是否显示最小化按钮。  $.fn.window.defaults.maximizable = false;//定义是否显示最大化按钮。  $.fn.window.defaults.shadow = false;//如果设置为true，在窗体显示的时候显示阴影。  $.fn.window.defaults.modal = true;//定义是否将窗体显示为模式化窗口。  $.fn.window.defaults.loadingMessage = '正在加载...';//在加载远程数据的时候在面板内显示一条消息。  } |

1. datagrid的默认值进行修改:

|  |
| --- |
| $.fn.datagrid.defaults.striped = true;//是否显示斑马线效果。  $.fn.datagrid.defaults.fit = true;//当设置为true的时候面板大小将自适应父容器。  $.fn.datagrid.defaults.loadMsg = '正在加载...';//在从远程站点加载数据的时候显示提示消息。  $.fn.datagrid.defaults.rownumbers = true;//如果为true，则显示一个行号列。  $.fn.datagrid.defaults.singleSelect = true;//如果为true，则只允许选择一行。  $.fn.datagrid.defaults.pageSize = 15;//在设置分页属性的时候初始化页面大小。  $.fn.datagrid.defaults.pageList = [15, 25, 50];//在设置分页属性的时候 初始化页面大小选择列表。  $.fn.datagrid.defaults.remoteSort = false;//定义从服务器对数据进行排序。  $.fn.datagrid.defaults.scrollbarSize = 10;//滚动条的宽度(当滚动条是垂直的时候)或高度(当滚动条是水平的时候)。 |

1. treegrid的默认值修改：

|  |
| --- |
| $.fn.treegrid.defaults.fit = true;//当设置为true的时候面板大小将自适应父容器。  $.fn.treegrid.defaults.loadMsg = '正在加载...';//在从远程站点加载数据的时候显示提示消息。  $.fn.treegrid.defaults.rownumbers = true;//如果为true，则显示一个行号列。  $.fn.treegrid.defaults.singleSelect = true;//如果为true，则只允许选择一行。  $.fn.treegrid.defaults.remoteSort = false;//定义从服务器对数据进行排序。  $.fn.treegrid.defaults.scrollbarSize = 10;//滚动条的宽度(当滚动条是垂直的时候)或高度(当滚动条是水平的时候)。 |

1. 重写按钮的样式:

|  |
| --- |
| $.fn.linkbutton.defaults.plain = true;//为true时显示简洁效果。 |

1. tabs 控件方法扩展，使用如下:

|  |
| --- |
| $('#id').tabs('refreshTabs'); //刷新当前tab页  $('#id').tabs('allTabs'); //获取当前tabs的所有页  $('#id').tabs('closeCurrent'); //关闭当前打开的tab页  $('#id').tabs('closeAll'); //关闭所有的tab页，设置了close属性的  $('#id').tabs('closeOther'); //关闭除去打开之外所有的tab页  $('#id').tabs('closeRight'); //关闭所有右边的tab页  $('#id').tabs('closeLeft'); //关闭所有左边的tab页 |

1. validatebox 扩展文本框验证规则，蓝色部分为扩充的验证，需要添加验证可以在这里添加，代码如下：

|  |
| --- |
| $.extend($.fn.validatebox.defaults.rules, {  //移动手机号码验证  mobile: {//value值为文本框中的值  validator: function (value) {  var reg = /^1[3|4|5|8|9]\d{9}$/;  return reg.test(value);  },  message: '输入手机号码格式不准确.'  },  phone: {  validator: function (value) {  var mobileRule = /^(13[0-9]|14[57]|15[0-9]|18[0-9]|17[678])\d{8}$/;  var telephoneRule = /^\d{3,4}-\d{7,8}$/;  return mobileRule.test(value) || telephoneRule.test(value);  },  message: '请输入手机号码或区号-座机号码'  }  }); |

1. validatebox 验证方法扩展，使用如下:

|  |
| --- |
| $('#id').validatebox('remove'); //删除验证  $('#id').validatebox('reduce'); //恢复验证 |

1. datagrid 获取正在编辑状态的行，使用如下:

|  |
| --- |
| $('#id').datagrid('getEditingRowIndexs'); //获取当前datagrid中在编辑状态的行编号列表 |

1. tabs 控件方法扩展，使用如下

|  |
| --- |
| $('#id').tabs('refreshTabs'); //刷新当前tab页  $('#id').tabs('allTabs'); //获取当前tabs的所有页  $('#id').tabs('closeCurrent'); //关闭当前打开的tab页  $('#id').tabs('closeAll'); //关闭所有的tab页，设置了close属性的  $('#id').tabs('closeOther'); //关闭除去打开之外所有的tab页  $('#id').tabs('closeRight'); //关闭所有右边的tab页  $('#id').tabs('closeLeft'); //关闭所有左边的tab页 |