**目 录**

[1 后台管理展示 1](#_Toc170142174)

[1.1 登录系统 1](#_Toc170142175)

[1.2 管理首页 1](#_Toc170142176)

[1.2.1 添加新闻 1](#_Toc170142177)

[1.2.2 管理新闻界面 2](#_Toc170142178)

[1.2.3 图片管理 2](#_Toc170142179)

[1.2.4 用户管理 3](#_Toc170142180)

[2 分页数据表的实现 3](#_Toc170142181)

[2.1 Layui前端框架的数据表功能 3](#_Toc170142182)

[2.1.1 导入框架的CSS样式和JS代码 5](#_Toc170142183)

[2.1.2 网页主体中写入表格 6](#_Toc170142184)

[2.1.3 实现表格的脚本代码 6](#_Toc170142185)

[2.2 后端数据接口的实现 7](#_Toc170142186)

[2.2.1 在后端路由表中添加/mydata路由 7](#_Toc170142187)

[2.2.2 定义返回JSON数据结构体 7](#_Toc170142188)

[2.2.3 路由功能的函数Mydata的实现 8](#_Toc170142189)

[2.3 最终效果 10](#_Toc170142190)

[3 其他说明 10](#_Toc170142191)

**1 后台管理展示**

对于一个网站来说，后台管理是必须且非常重要的一个功能，主要实现网站数据维护、用户维护等功能。

**1.1 登录系统**



图1.1 登录系统

**1.2 管理首页**

登录系统后即出现管理首页，主要管理功能包括：用户管理、图片管理、新闻（文章）管理等。

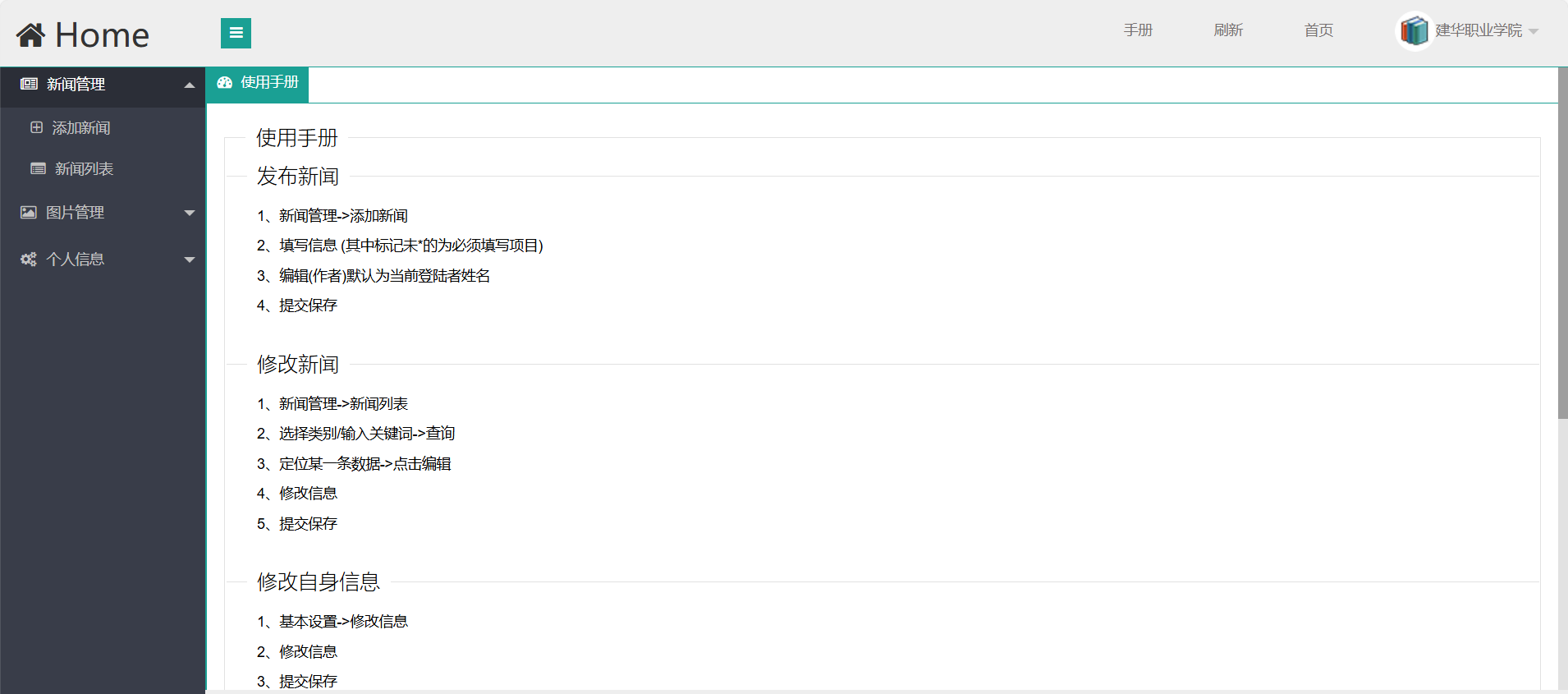


图1.2 后台管理主界面

**1.2.1 添加新闻**

添加新闻功能实现新闻的添加工作，界面主要以输入框的方式提供输入功能，最后提交数据即将输入的数据保存到数据库中。

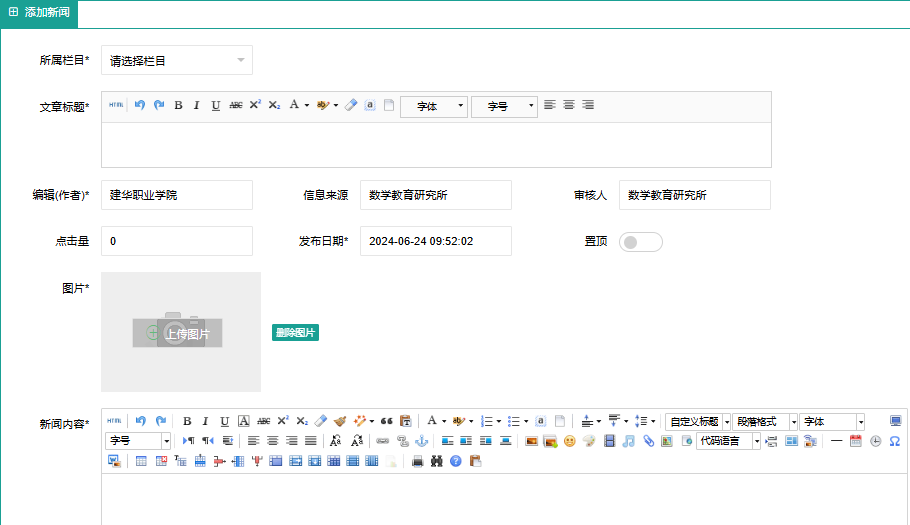


图1.3 添加新闻界面

**1.2.2 管理新闻界面**

新闻管理功能主要实现新闻的修改、删除功能。

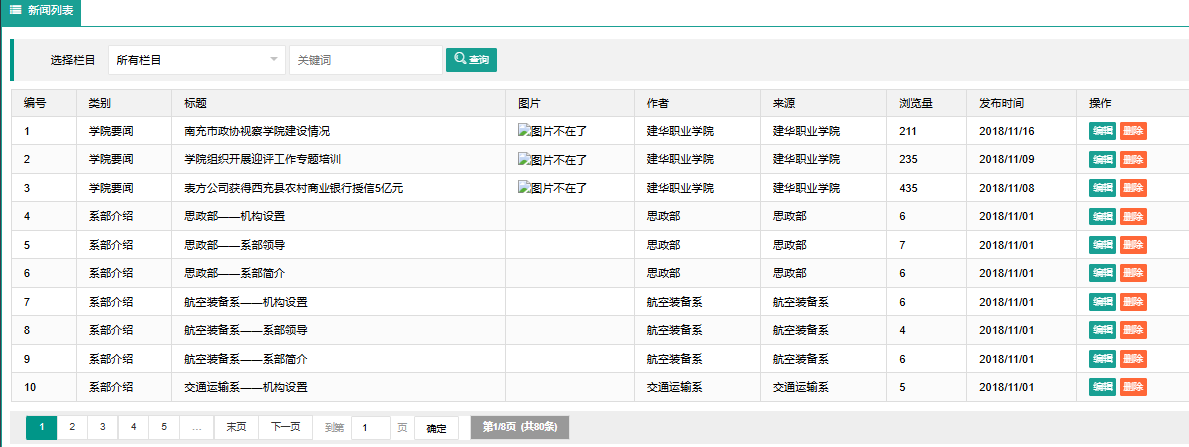


图1.4 新闻管理界面

**1.2.3 图片管理**

该功能主要实现对网站首页轮播图的管理（增、删、改）。



图1.5 添加图片界面



图1.6 图片管理界面

**1.2.4 用户管理**

该功能主要实现对用户的增删查改工作，包括设置用户权限、密码等。



图1.7 修改密码界面

**2 分页数据表的实现**

在进行新闻管理时，肯定是以分页方式逐页展示数据库中的数据信息，并在每条信息的后面显示“编辑”和“删除”两个按钮，下面以Layui前端框架为例展示分页数据表的实现方法。



图2.1 分页显示效果

**2.1 Layui前端框架的数据表功能**

Layui框架的组件中实现了分布数据表功能，而且实现样式多、功能全、操作简便。查看示例网址：<https://layui.dev/docs/2/table/#examples>

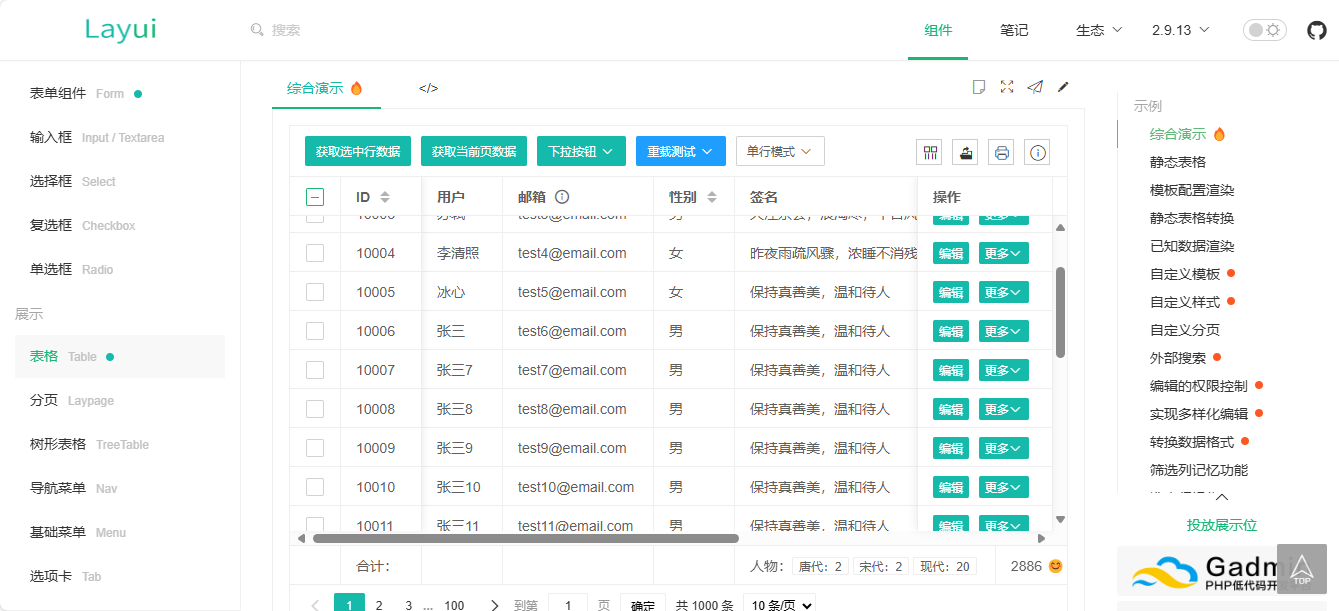


图2.2 Layui的数据表

使用Layui框架只需一个<table>标签再加上简单的表格接口代码即可实现后端数据到前端的分页展示效果，其中前端表格样例如下：

<!DOCTYPE html>

<html>

<head>

    <meta charset="utf-8">

    <title>后台管理</title>

    <link rel="stylesheet" href="/static/layui/css/layui.css" media="all">

    <script src="/static/layui/layui.js"></script>

</head>

<body>

    <table id="demo" lay-filter="test"></table>

    <script>

        layui.use('table', function () {

            var table = layui.table;

            table.render({

                elem: '#demo'

                , height: 500

                , url: '/mydata'    //数据接口

                , page: true        //开启分页

                , method: "post"

                , cols: [[          //表头

                    { field: 'Id', title: '序号', width: 100, sort: true, fixed: 'left', align: "center" }

                    , { field: 'Xuehao', title: '学号', width: 200, align: "center", sort: true }

                    , { field: 'Xingming', title: '姓名', width: 150, sort: true, align: "center" }

                    , { field: 'Ban', title: '年级、班', width: 150 }

                    , { field: 'Banma', title: '班级代码', width: 200 }

                    , { field: 'Kecheng', title: '课程名称', width: 80, sort: true }

                    , { field: 'Zy', title: '专业名称', width: 200, sort: true }

                ]],

                parseData: function (res) {

                    return {

                        "code": res.code,           //返回代码

                        "data": res.data,           //返回数据

                        "count": res.count,         //记录总数

                        "msg": res.msg,             //错误信息

                        "status": res.status        //返回状态

                    }

                }

            });

        });

    </script>

</body>

</html>

**2.1.1 导入框架的CSS样式和JS代码**

<link rel="stylesheet" href="/static/layui/css/layui.css" media="all">

<script src="/static/layui/layui.js"></script>

本例直接下载前端框架到本地文件夹中，网站运行时以绝对路径访问资源内容即以/开始的路径去访问资源。

**2.1.2 网页主体中写入表格**

<table id="demo" lay-filter="test"></table>

根据你的实际需要设置表格id，上例中设定的表格id为“demo”。

**2.1.3 实现表格的脚本代码**

上面示例中<script>…</script>中的代码即中表格实现代码，主要需要设置的是关联表格的id、表格高度、数据接口路由、是否分页、数据发送方式、表头设置以及后端数据解析等。

**（1）关联表格id**

其中的“elem:”设置关联表格的id，只需设置为与前面2.1.2节中你的表格id相同即可。但是注意这里的id前面要加上“#”。

**（2）设置表格高度**

表格高度一般是与你显示的每页信息条数直接相关的，若高度小于实际需要的高度则表格右边会自动显示垂直滚动条，高度设置“height:”后面直接给定数值即可，不用单位，默认为像素。

**（3）数据接口路由**

“url:”后面给定数据接口路由，通过这个路由对应的函数去实现查询数据库中数据并返回数据然后以表格方式显示出来的功能，Layui实现了前端显示功能，我们只需要实现该路由的数据查询并返回数据的功能即可。

**！！！特别注意**：后端查询到的数据必须以JSON格式返回。

**（4）是否分页**

“page:”设置是否分页，true表示以分页方式展示数据，false表示不分页。

**（5）数据提交方式**

也就是以什么样的方式提交你要访问的第几页（page）以及每页信息条数（limit）两个数据给接口路由，在接口路由函数中可以直接接收这两个数据，以便确定提取第几页的数据，提取多少。提交方式有两种，一是post，二是get。

**（6）表头数据格式**

在“cols:”中指定表格的基本样式，包括每列的表头显示名称，宽度、对齐方式等等。其中“field”指定接收到的后端数据表数据的列名称，与后端程序中定义的列名称一致，注意大小写；“title”指定列表头显示名称；“width”指定列宽，直接给数字，不需要带单位，默认为像素；“sort”指定是否允许按列排序，如果允许，显示的列名称旁边会显示排序按钮，可以完成单列排序功能，指定值为true表示允许排序，false表示不能排序，默认为false，如图2.3所示；“align”指定列数据对齐方式，可以指定值为center、left、right等。



图2.3 排序列表头

**（7）数据解析**

“parseData”指定数据解析方式，也就是接收到后端传递过来的JSON数据格式是什么，一般要求后端传过来的JSON数据包括code、data、count、msg、status等字段信息。其中，code表示返回的数据类别代码；data表示返回的一页表格数据；count表示表格中记录总条数；msg为错误提示信息，即当返回不成功时的提示信息；status为状态码，一般200表示正确返回。

**2.2 后端数据接口的实现**

从2.1节已经知道，Layui框架实现数据表功能只需要后面实现一个数据接口路由并以JSON格式返回提取到的表达数据即可。

接2.1中的前端程序示例，其数据接口路由为“/mydata”，因此后端需要实现该路由的功能，即提取指定页（page）和指定条数（limit）的数据并以JSON格式返回，因此，后端程序需要做以下几件事情：

（1）在路由表中添加该路由对应的函数，注意提交方法与前端指定的方法一致；

（2）定义返回的JSON格式结构体；

（2）实现路由函数，在函数中完成数据提取功能并返回数据。

**2.2.1 在后端路由表中添加/mydata路由**

beego.Router("/mydata", &controllers.MainController{}, "post:Mydata")

本例直接在router.go中添加，路由函数为Mydata，提交方法为post。

**2.2.2 定义返回JSON数据结构体**

由于前端参数接收以及数据返回均在控制器中完成，因此，在控制器中添加JSON数据结构体，本例直接在default.go中添加一个结构体，形如：

type PageData struct {

    Code   int         `json:"code"`

    Data   interface{} `json:"data"`

    Count  int64       `json:"count"`

    Msg    string      `json:"msg"`

    Status int         `json:"status"`

}

结构体取名为PageData，也可以根据实际需要任意取名，但是尽量做到见名知义。结构体中定义了用于前端接收的数据字段。

**2.2.3 路由功能的函数Mydata的实现**

该函数实现/mydata路由并返回JSON数据，首先接收前端发送过来的两个分页数据page和limit，其中page指定当前页，limit指定每页记录数；然后访问数据库提取对应的数据，同时提取到数据库总记录数；最终生成JSON数据并返回。

**（1）提取数据功能的实现**

数据库操作函数放在models中。因此在models的对应表格文件中添加一个提取数据的功能函数，本例读取学生信息表数据即stud表数据，因此在stud.go中添加函数，设计函数名为GetPageData，函数参数为当前页为每页条数，函数返回提取到的总记录数、若干条数据（结构体切片）以及是否成功提取等，具体实现如下：

func GetPageData(page, limit int) (int64, []Stud, bool) {

    var e error

    var r int64

    //（1）获取总记录数

    d1 := new(Stud)

    r, e = Db.Count(d1)

    if e != nil {

        return 0, nil, false

    }

    //(2)读取记录

    d2 := make([]Stud, 0)

    e = Db.Limit(limit, (page-1)\*limit).Find(&d2)

    if e != nil {

        return 0, nil, false

    } else {

        return r, d2, true

    }

}

**（2）路由函数Mydata的实现**

路由函数主要完成接收前端数据，访问models中的GetPageData函数提取数据，生成JSON数据并返回。具体实现如下：

// 数据表接口会传递两个参数过来

// 参数1：page，当前页号（从1开始）

// 参数2：limit，每页记录数

// 返回：JSON数据

func (c \*MainController) Mydata() {

    //(1) 获取传递参数

    limit, \_ := c.GetInt("limit")

    page, \_ := c.GetInt("page")

    //(2) 读数据

    r, d, e := models.GetPageData(page, limit)

    //(3) 生成JSON数据

    if !e {

        c.Data["json"] = &PageData{0, nil, 0, "读取数据错误！", 400}

    } else {

        c.Data["json"] = &PageData{0, d, r, "成功", 200}

}

    //(4) 返回JSON

    c.ServeJSON()

}

由于要返回JSON格式数据，因此使用“c.Data["json"]”设置要返回的数据，而且最后使用“c.ServeJSON()”直接返回，而一般页面返回是使用“c.TplName”实现的。

**2.3 最终效果**

将前面项目登录管理后台的页面admin.html的内容改为2.1节中的内容，正确登录管理后台，显示效果如下：



图2.4 实例效果

默认分页大小为每页10条数据，点击表格正文的分页数字可以显示对应的页的数据，也可以通过后面的下拉列表选择每页记录数。

**3 其他说明**

管理后台的设计和实现涉及到很多技术，如加密技术、防止CSRF技术、Session技术、Cookie技术等。另外，在实现数据输入时一般还要用到富文本编辑器（RTF Editor），常用的如TinyMCE（https://www.tiny.cloud/）、CKEditor（https://ckeditor.com/）、UEditor（https://github.com/fex-team/ueditor）等，直接搜索“富文本编辑器”即可查阅到很多相关资料。