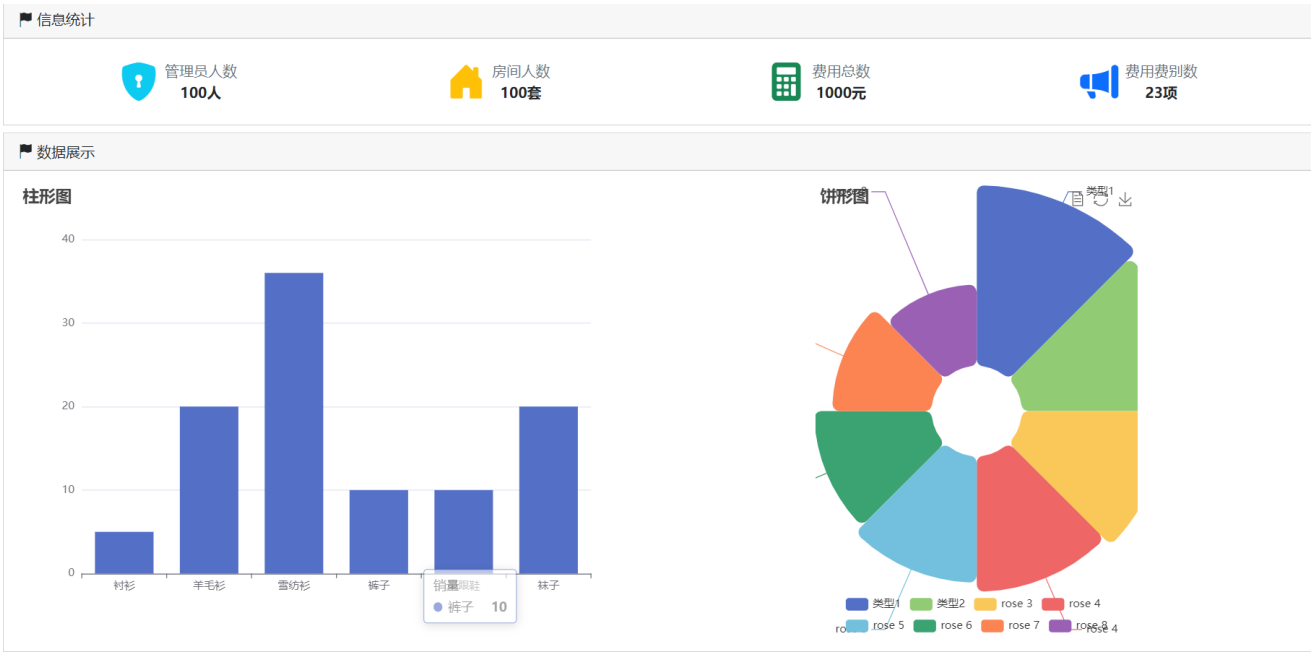
数据可视化echarts操作

效果图



一、界面完成

关于界面布局，我们就不再讲了，现在先讲一下Service

二、DataViewService的完成

因为这个功能不涉及到任何一个具体的具，所以我们在`service`的文件夹下面新建了一个Service叫`DataViewService`代码如下

```
/**
 * 用于数据展示的Service操作，不涉及到任何的具体的表
 */
const BaseService = require("./BaseService");

class DataViewService extends BaseService {
    constructor() {
        super();
        //因为不涉及到任何具体的表，所以我不用传参给父级，这样我们也不使用
currentTableName这个属性
    }

    /**
     * 获取计算的总数
     * @returns {Promise<Object>} 返回查询的结果的对象
     */
    async getCalcData() {
```

```
        let strSql = `select
         max(if(adminCount='adminCount',totalCount,0)) 'adminCount',
         max(if(adminCount='roomCount',totalCount,0)) 'roomCount',
max(if(adminCount='moneyCount',totalCount,0)) 'moneyCount',
max(if(adminCount='costTypeCount',totalCount,0)) 'costTypeCount'
from(select 'adminCount',count(*) 'totalCount' from
${this.tableMap.admininfo}
        union allselect 'roomCount',count(*) 'totalCount' from
${this.tableMap.roominfo}
        union all        select 'moneyCount',sum(totalmoney) 'totalMoney'
from ${this.tableMap.costinfo}
        union all        select 'costTypeCount', count(*) 'totalCount' from
${this.tableMap.costtype}) a`;
        let result = await this.executeSql(strSql);
        return result[0];
    }
}


module.exports = DataViewService;
```

> 在上面的代码里在，我们可以看到使用了`union all`来进行结果集并联，也使用了行转列的操作，这个sql语句执行的结果如所示
>
> 信息 结果 1 剖析 状态
>
adminCount	roomCount	moneyCount	costTypeCount
32.00	73.00	160214.79	44.00

当我们把Service完成了以后，我们就要再进入路由操作

三、在工厂里面生产这个Service

```
/**
 * @author 杨标
 * @description 服务层工厂
 */

class ServiceFactory {
    static createAdminInfoService() {
        const AdminInfoServcie = require("../services/AdminInfoService");
        return new AdminInfoServcie();
    }

    static createRoomInfoService() {
        const RoomInfoService = require("../services/RoomInfoService");
        return new RoomInfoService();
    }

    static createCostTypeService() {
```

```
        const CostTypeService = require("../services/CostTypeService");
        return new CostTypeService();
    }

    static createCostInfoService() {
        const CostInfoService = require("../services/CostInfoService");
        return new CostInfoService();
    }

    static createDataViewService(){
        const DataViewService = require("../services/DataViewService");
        return new DataViewService();
    }
}

module.exports = ServiceFactory;
```

四、完成路由dataViewRouter.js

```
/**
 * @author 杨标
 * @description dataView的路由模块
 */
const express = require("express");
const router = express.Router();


module.exports = router;
```

当我们创建好路由文件以后，一定要在`app.js`里面链接我们的路由文件，这样才会有一个一级路径

```
app.use("/dataView",require("./routes/dataViewRouter"));
```

当我们把所有的工作都准备好了以后，我们可以在`dataViewRouter.js`里面来处理我们的请求了

```
/**
 * @author 杨标
 * @description dataView的路由模块
 */
const express = require("express");
const router = express.Router();
const ServiceFactory = require("../factory/ServiceFactory");
const ResultJson = require("../model/ResultJson");

router.get("/getCalcData", async (req, resp) => {
```

```
    let result = await
ServiceFactory.createDataViewService().getCalcData();
    let resultJson = new ResultJson(Boolean(result), result ? "获取数据成功"
: "获取数据失败", result);
    resp.json(resultJson);
});


module.exports = router;
```

现在的前端页面只在请求这一个地址就可以获取到数据了

```
<script>
    $(function () {
        async function getCalcData() {
            try {
                let result = await
request.get(`${baseURL}/dataView/getCalcData`);
                $("#adminCount").text(result.data.adminCount);
                $("#costTypeCount").text(result.data.costTypeCount);
                $("#moneyCount").text(result.data.moneyCount);
                $("#roomCount").text(result.data.roomCount);
            } catch (error) {
                console.log(error)
            }
        }

        getCalcData();
    })
</script>
```

五、百度图形图表的使用

1.图形图表的介绍

> 图形图表用一个前端专业的术语来说叫数据可视化，它是将一系列的数据转换成可见化的展示操作

目前能够实现数据可视化的框架有很多，但使用的最得最多的就是以下几种

- `hicharts`这个是国外使用得比较多的一个框架，这个框架功能强大，非常好用，但是收费
- `echarts`这是百度为了我们推出的一个图形图表框架，完全免费，并且已经将这个框架捐献给了`apache`开源组织。它实现了`hicharts`中99%的功能
  Apache ECharts
- `datav`阿里巴大数据展示库，但是这个也是个人免费，企业需要购买授权

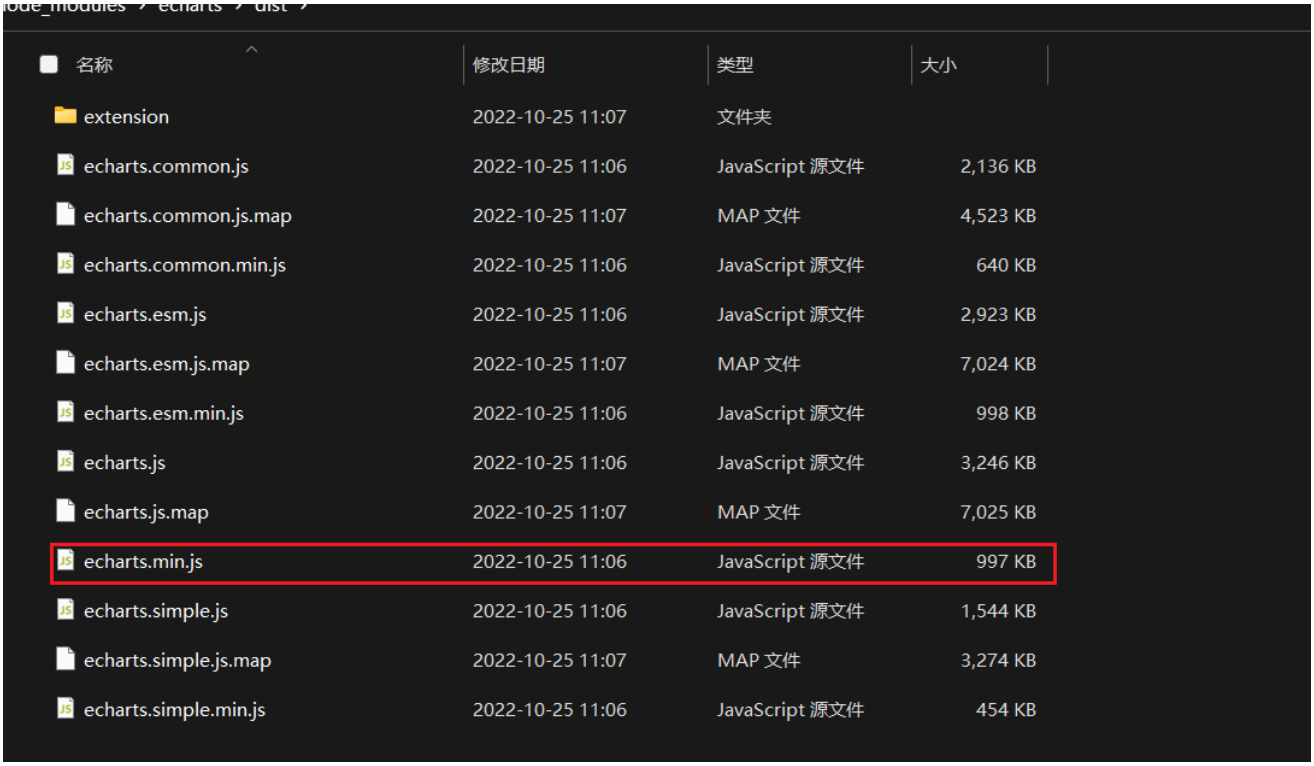
综合对比以后，其实现在的企业多数使用的都是百度的echarts来完成数据展示

2.百度图表的安装与导入

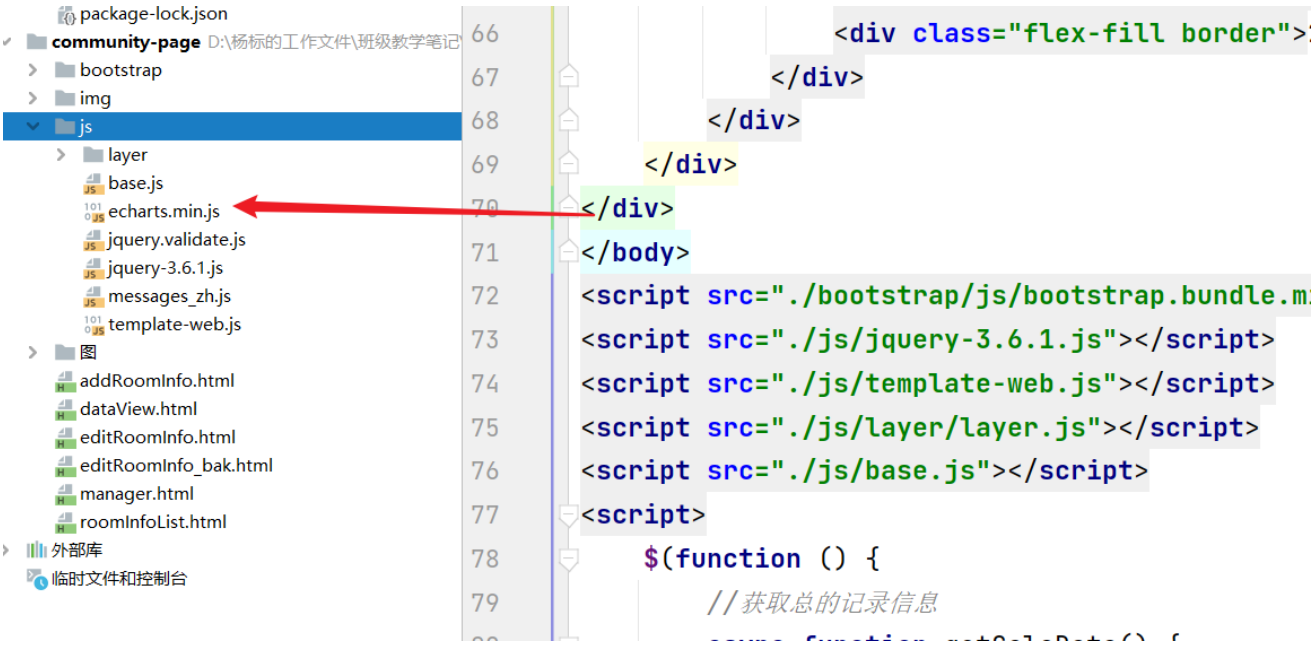
下载 - Apache ECharts
上面就是百度图表的下载地址，我们可以通过`github`来进行下载，也可以通过`npm`仓库来进行下载

```
$ npm install echarts
```

随便找一下空的目录执行上面的命令，这个时候`ecahrts`就会从服务器下载下来，我们从`node_modules`里面找到一个文件`echarts.min.js`这个文件，它就是我们所需要用到的文件



把这个文件复制到自己的`js`的目录下面来



同时，我们要把这个文件导入到所需要的页面，这里我们导入的是`dataView.html`这个页面

```
<script src="./js/echarts.min.js"></script>
```

3.百度echarts的使用

```
function initChart1(){
    // 基于准备好的dom，初始化echarts实例  将id="chart1"的这个区域初始化为图表的展示区域
    var myChart = echarts.init(document.getElementById('chart1'));
    // 指定图表的配置项和数据
    var option = {
        title: {
            text: '费用类别'
        },
        tooltip: {},
        legend: {
            data: ['费用总数']
        },
        xAxis: {
            data: ['电费', '水费', '保护费', '清洁费', '煤气费', '物业费']
        },
        yAxis: {},
        series: [
            {
                name: '费用总数',
                type: 'bar',
                data: [50, 20, 36, 10, 10, 20]
            }
        ]
    };
    // 使用刚指定的配置项和数据显示图表。  这个方法就是用于设置，怎么样去展示图形图表
    myChart.setOption(option);
```

```
    myChart.setOption(option);
}
initChart1();
```

当我们把所有的图表都设置完成以后，我们就想着办法去从后台服务器获取真实的数据，这里我们以类用类别来汇总，进行费总的统计信息

4.从后台获取数据渲染图表

DataViewService.js

```
/**
 * 用于数据展示的Service操作，不涉及到任何的具体的表
 */
const BaseService = require("./BaseService");

class DataViewService extends BaseService {
    constructor() {
        super();
        //因为不涉及到任何具体的表，所以我不用传参给父级，这样我们也不使用
currentTableName这个属性
    }

    /**
     * 获取计算的总数
     * @returns {Promise<Object>} 返回查询的结果的对象
     */
    async getCalcData() {
        let strSql = `select
         max(if(adminCount='adminCount',totalCount,0)) 'adminCount',
         max(if(adminCount='roomCount',totalCount,0)) 'roomCount',
max(if(adminCount='moneyCount',totalCount,0)) 'moneyCount',
max(if(adminCount='costTypeCount',totalCount,0)) 'costTypeCount'
from(select 'adminCount',count(*) 'totalCount' from
${this.tableMap.admininfo}
        union allselect 'roomCount',count(*) 'totalCount' from
${this.tableMap.roominfo}
        union all        select 'moneyCount',sum(totalmoney) 'totalMoney'
from ${this.tableMap.costinfo}
        union all        select 'costTypeCount', count(*) 'totalCount' from
${this.tableMap.costtype}) a`;
        let result = await this.executeSql(strSql);
        return result[0];
    }

    /**
     * 获取费用类别的总金额
     * @return {Promise<Array>}
     */
```

```
     */
    async getCostTypeTotalMoney() {
        let strSql = `select a.costname,sum(a.totalmoney) 'totalMoney' from
(select a.*,b.costname from ${this.tableMap.costinfo} a             inner
join ${this.tableMap.costtype} b on a.costid = b.id) a            group by
a.costid`;
        let result = await this.executeSql(strSql);
        return result;
    }
}


module.exports = DataViewService;
```

dataViewRouter.js路由文件

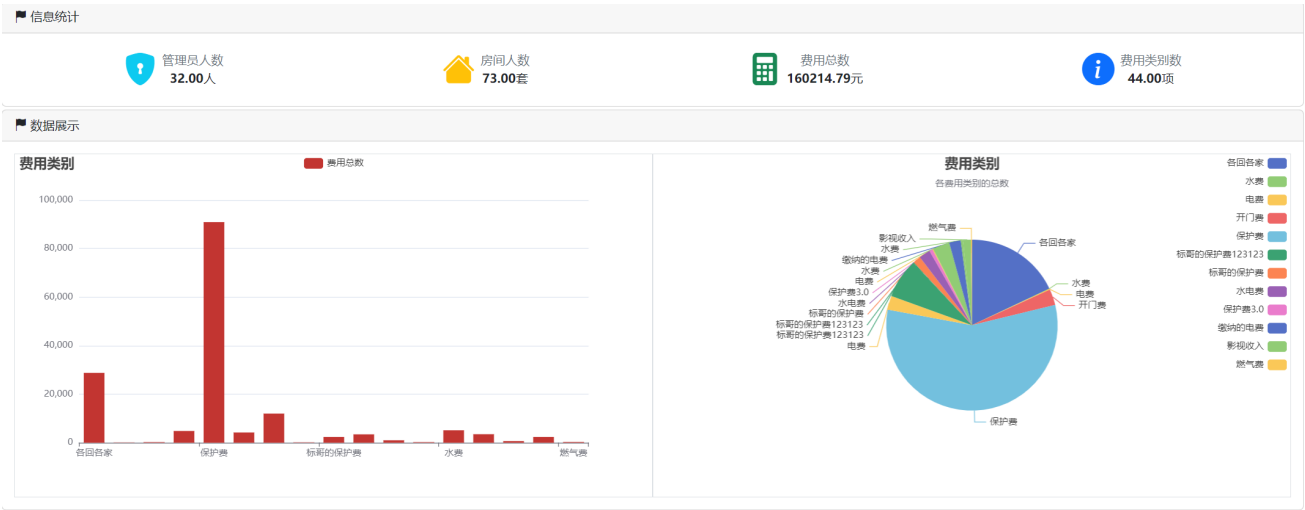
```
router.get("/getCostTypeTotalMoney",async(req,resp)=>{
    let result = await
ServiceFactory.createDataViewService().getCostTypeTotalMoney();
    let resultJson = new ResultJson(true,"获取数据成功",result);
    resp.json(resultJson);
})
```

当所有的后台功能完成了以后，我们就可以在页面上面请求这些接口了

```
//初始化第一个图形图表
async function initChart() {
    // 第一步： 基于准备好的dom，初始化echarts实例  将id="chart1"的这个区域初始化为
图表的展示区域
    let myChart1 = echarts.init(document.getElementById('chart1'));
    // 指定图表的配置项和数据
    let option1 = {
        title: {
            text: '费用类别'
        },
        tooltip: {},
        legend: {
            data: ['费用总数']
        },
        xAxis: {
            // data: ['电费', '水费', '保护费', '清洁费', '煤气费', '物业费']
            data: []
        },
        yAxis: {},
        series: [
            {
                name: '费用总数',
                type: 'bar',
```

```
            // data: [50, 20, 36, 10, 10, 20]
            data: [],
        }
    ],
    color: "#c23531"
};

//----------------------第二个图表---------------------------------
//第一步：初始化
let myChart2 = echarts.init(document.getElementById('chart2'));
//第二步：配置option
let option2 = {
    title: {
        text: '费用类别',
        subtext: '各费用类别的总数',
        left: 'center'
    },
    tooltip: {
        trigger: 'item'
    },
    legend: {
        orient: 'vertical',
        left: 'right'
    },
    series: [
        {
            name: 'Access From',
            type: 'pie',
            radius: '50%',
            data: [
                // {value: 1048, name: 'Search Engine'},
                // {value: 735, name: 'Direct'},
                // {value: 580, name: 'Email'},
                // {value: 484, name: 'Union Ads'},
                // {value: 300, name: 'Video Ads'}
            ],
            emphasis: {
                itemStyle: {
                    shadowBlur: 20,
                    shadowOffsetX: 20,
                    shadowColor: 'rgba(0, 0, 0, 0.5)'
                }
            }
        }
    ]
};


//第三步：请求后台的数据
```

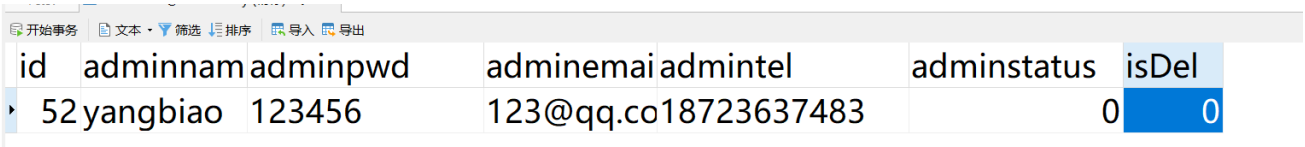
```
    try {
        let result = await
request.get(`${baseURL}/dataView/getCostTypeTotalMoney`);
        result.data.forEach(item => {
            option1.xAxis.data.push(item.costname);
            option1.series[0].data.push(item.totalMoney);

            //处理第二个表格的数据
            option2.series[0].data.push({
                value:item.totalMoney,
                name:item.costname
            })
        });
        // 使用刚指定的配置项和数据显示图表。    这个方法就是用于设置，怎么样去展示图形
图表
        // 第四步:设置这一个option
        myChart1.setOption(option1);
        myChart2.setOption(option2);

    } catch (error) {
        console.log(result);
    }
}

initChart();
```



> 后期如果再需要其它的图形图表，只要参照示例去完成就可以了

六、管理员的模块

管理员模块也是一个增删更查的操作，但是它里有一点与别人不一样，它会涉及到密码的操作



id	adminnam	adminpwd	adminemai	admintel	adminstatus	isDel
52	yangbiao	123456	123@qq.co	18723637483	0	0

当我们向数据库里面存储上面的记录的时候，我们可以看到，用户的密码实现的是明文存储。只要有人拿到这个数据表，就可以看到所有的密码
正常情况下的密码应该都是加密的，如`md5`加密

AdminInfoService.js

```
/**
 * @author 杨标
 * @description admininfo模块的操作
 */

const BaseService = require("./BaseService");

class AdminInfoService extends BaseService {
    constructor() {
        super("admininfo");
    }

    /**
     * 新增管理员信息
     * @param {{adminname, adminpwd, adminemail, admintel, adminstatus}}
     * @return {Promise<boolean>} true代表新增成功,false代表新增失败
     */
    async add({adminname, adminpwd, adminemail, admintel, adminstatus}) {
        let strSql = `INSERT INTO ${this.currentTableName} (adminname,
adminpwd, adminemail, admintel, adminstatus) VALUES (?, ?, ?, ?, ?);`
let result = await this.executeSql(strSql, [adminname, adminpwd,
adminemail, admintel, adminstatus]);
        return result.affectedRows > 0;
    }
}
module.exports = AdminInfoService;
```

adminInfoRouter.js

```
/**
 * @author 杨标
 * @description adminInfo路由模块
 */
const express = require("express");
const router = express.Router();
const ServiceFactory = require("../factory/ServiceFactory");
const ResultJson = require("../model/ResultJson");

router.post("/addAdminInfo", async (req, resp) => {
```

```
router.post("/addAdminInfo", async (req, resp) => {
    //当我们接收到前台传递过来的参数以的,我们就要插入到数据库
    let result = await
ServiceFactory.createAdminInfoService().add(req.body);
    let resultJson = new ResultJson(result, result ? "新增成功" : "新增失败");
    resp.json(resultJson);
});

module.exports = router;
```

经过上面的操作以后，我们后台代码已经完在了，现在的页面上面只要提交数据就可以保存到后台的数据库里面

addAdminInfo.html

```
<!DOCTYPE html>
<html lang="zh">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>新增管理员</title>
    <link rel="stylesheet" href="./bootstrap/css/bootstrap.min.css">
    <link rel="stylesheet" href="./bootstrap/font/bootstrap-icons.css">
    <link rel="stylesheet" href="./js/layer/theme/default/layer.css">
</head>

<body>
<div class="container-fluid">
    <ul class="breadcrumb my-2">
        <li class="breadcrumb-item"><a href="dataView.html">首页</a></li>
        <li class="breadcrumb-item active">新增管理员</li>
    </ul>
</div>
<div class="container">
    <div class="h4 text-center text-primary border-bottom py-3">新增管理员
</div>
    <form id="addAdminInfoForm">
        <div class="form-floating mt-3 mb-3">
            <input type="text" class="form-control"
                   placeholder="管理员姓名" id="adminname" name="adminname"
                   data-rule-required="true" data-msg-required="管理员姓名不
能为空">
            <label for="adminname">管理员姓名</label>
        </div>
        <div class="form-floating mt-3 mb-3">
            <input type="text" class="form-control"
                   placeholder="管理员密码" id="adminpwd" name="adminpwd"
                   data-rule-required="true" data-msg-required="管理员密码不
```

```
                    data-rule-required="true" data-msg-required="管理员密码不
能为空">
            <label for="adminpwd">管理员密码</label>
        </div>
        <div class="form-floating mt-3 mb-3">
            <input type="text" class="form-control"
                   placeholder="请输入确认密码" id="confirmPwd"
name="confirmPwd"
                   data-rule-required="true" data-msg-required="确认密码不能
为空"
                   data-rule-equalTo="#adminpwd" data-msg-equalTo="两次密码必
须相同">
            <label for="confirmPwd">确认密码</label>
        </div>
        <div class="form-floating mt-3 mb-3">
            <input type="text" class="form-control"
                   placeholder="管理员邮箱" id="adminemail" name="adminemail"
                   data-rule-required="true" data-msg-required="管理员邮箱不
能为空"
                   data-rule-regexp="\w+([-+.]\w+)*@\w+([-.]\w+)*\.\w+
([-.]\w+)*" data-msg-regexp="请输入正确的邮箱格式">
            <label for="adminemail">管理员邮箱</label>
        </div>
        <div class="form-floating mt-3 mb-3">
            <input type="text" class="form-control"
                   placeholder="请输入手机号码" id="admintel" name="admintel"
                   data-rule-required="true" data-msg-required="手机号不能为
空"
                   data-rule-regexp="^(0|86|17951)?(13[0-
9]|15[012356789]|166|17[3678]|18[0-9]|14[57])[0-9]{8}$"
                   data-msg-regexp="请输入正确的手机号">
            <label for="admintel">手机号码</label>
        </div>
        <div class="form-floating mt-3 mb-3">
            <select class="form-select" name="adminstatus"
id="adminstatus">
                <option value="0">正常</option>
                <option value="1">禁用</option>
            </select>
            <label for="adminstatus">管理员状态</label>
        </div>
        <div class=" d-flex">
            <button type="button" class="btn btn-primary" id="btn-save">
                <i class="bi bi-file-pdf"></i>
                保存
            </button>
            <button type="button" class="btn btn-warning mx-3">
                <i class="bi bi-arrow-90deg-down"></i>
                重置

            </button>
```

```
            </button>
        </div>
    </form>
</div>
</body>
<script src="./bootstrap/js/bootstrap.bundle.min.js"></script>
<script src="./js/jquery-3.6.1.js"></script>
<script src="./js/template-web.js"></script>
<script src="./js/layer/layer.js"></script>
<script src="./js/jquery.validate.js"></script>
<script src="./js/messages_zh.js"></script>
<script src="./js/base.js"></script>
<script>
    $(function () {
        $.validator.addMethod("regexp", function (value, element, params) {
            var reg = new RegExp(params);
            return reg.test(value);
        });
        //表单验证
        var addAdminInfoFormResult = $("#addAdminInfoForm").validate({
            errorPlacement: function (error, element) {
                // error代表的就是这个错误提示的消息
                // element代表你当前正在验证的这个元素
                element.parent().after(error);
            },
            errorClass: "text-danger"
        });

        $("#btn-save").click(async function () {
            //第一步:做表单验证
            if (addAdminInfoFormResult.form()) {
                //验证通过,我们就将数据通过ajax发送到后台
                try {
                    let result = await
request.post(`${baseURL}/adminInfo/addAdminInfo`, {
                        adminname: $("#adminname").val(),
                        adminpwd: $("#adminpwd").val(),
                        adminemail: $("#adminemail").val(),
                        admintel: $("#admintel").val(),
                        adminstatus: $("#adminstatus").val()
                    });
                    layer.alert("新增成功");
                } catch (error) {
                    layer.alert("服务器错误");
                    console.log(error);
                }
            } else {
                layer.alert("请检测你的输入信息...");
            }

        })
```

```
        })
    })
</script>

</html>
```

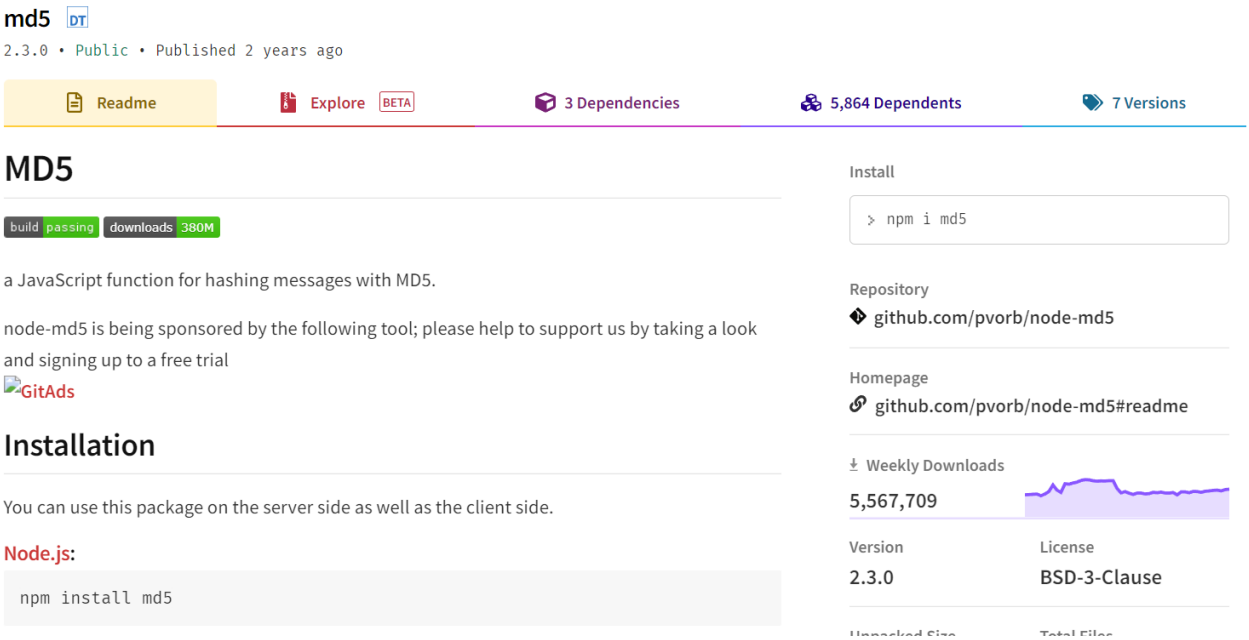
开始事务 | 文本 | 筛选 | 排序 | 导入 | 导出

id	adminnam	adminpwd	adminemai	admintel	adminstatus	isDel
53	杨标	123456	lovesnsfi@	18627193876	0	0

这个时候的数据库已经可以存储我们的数据了，但是仍然要注意，这个时候的密码是明文存储的，很危险，我们要对密码 `adminpwd` 进行加密

七、md5加密

为了解决密码的存储问题，我们要对密码这一个字段进行加密，加密的方式们采用 `md5`，在 `nodejs` 的平台下面，有一个第三方模块可以完成这个操作



```
const BaseService = require("./BaseService");
const md5 = require("md5");

class AdminInfoService extends BaseService {
    constructor() {
        super( currentTableName: "admininfo");
    }

    /**
     * 新增管理员信息
     * @param {{adminname, adminpwd, adminemail, admintel, adminstatus}}
     * @return {Promise<boolean>} true代表新增成功,false代表新增失败
     */
    async add({adminname, adminpwd, adminemail, admintel, adminstatus}) {
        //现在在这里,要执行sql语句的新增之前,我们要把密码进行md5的加密
        adminpwd = md5(adminpwd);
        let strSql = `INSERT INTO ${this.currentTableName} (adminname, adminpwd, adminemail, admintel, adminstatus) VALUE
        let result = await this.executeSql(strSql, params: [adminname, adminpwd, adminemail, admintel, adminstatus]);
        return result.affectedRows > 0;
    }
}
```

在这里，我们就把密码进行了一次加密操作

对象 admininfo @community (杨标) - 表

开始事务 文本 筛选 排序 导入 导出

id	adminnam	adminpwd	adminemai	admintel	adminstatus	i
54	杨标	e10adc3949ba59abbe56e057f20f883e	lovesnsfi@	18627193876	0	

这个时候当我们再次去新增的时候，我们发现密码就已经不再是原来的明文存储了，而是像乱码一样的存储，这就是md5

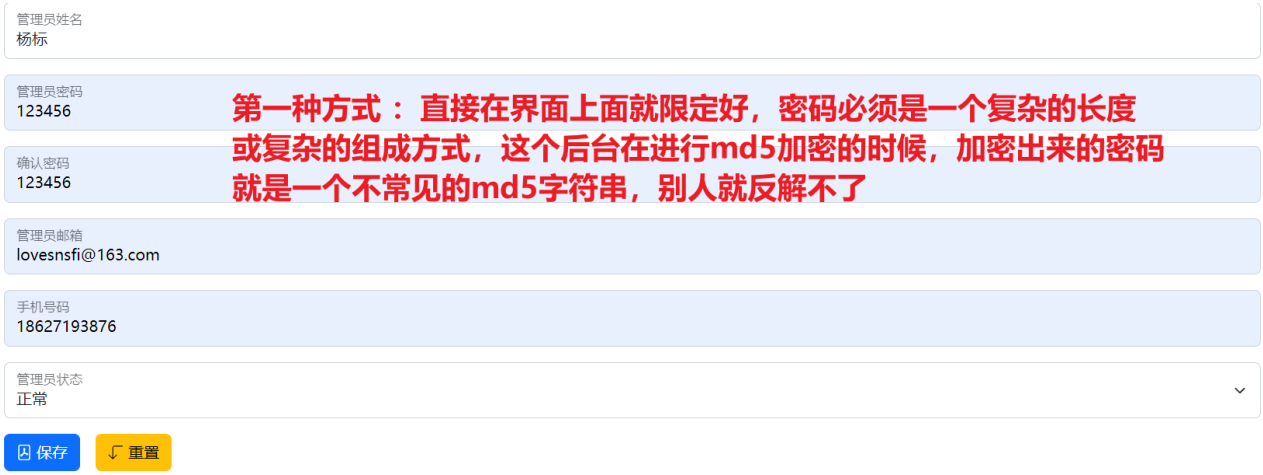
到这一步为止，我们就已民经完成了md5密码的加密操作

问题：现在有一个非常大的问题，当我们拿着这个加密以后的字符串去cmd5的网站去解密的时候，发现这个密码竟然可以反解密出来。如下所示

这样就非常不安全，怎么办？

> **md5的加密是不可破解的，为什么cmd5的网站可以解密出来是因为123456太常见了，这个网站就一个一个的去试，它试出来。**如果我们要是把密码强大改大一点，这样别人就破解不了

第一种常见的解决方法



还有一种解决方案就是md5加密+加盐

八、md5加盐

上面的问题已经展示得非常清，我们在这里如果进行md5的简单值加密的时候是会破解的，我们在得到用户的密码以后，可以手动的在用户的密码后面或前面添加一个特殊的字符，以保证加密的字符串的不规则性

第一步：我们在 `config` 目录下面的AppConfig下面配置的

```
/**
 * 整个程序的配置文件，所以需要配置的东西，我都写在这里
 * @author 杨标
 * @date 2022-10-24
 */const path = require("path");

const AppConfig = {
    excelDir: path.join(__dirname, "../excelDir"),
    salt: "098lskdf.!@#09sdfj"
}

module.exports = AppConfig;
```

第二步：在加密的时候，我们在原来用户输入的密码上面，添加了盐的值

```
const md5 = require("md5");
const AppConfig = require("../config/AppConfig");

class AdminInfoService extends BaseService {
    constructor() {
        super( currentTableName: "admininfo");
    }

    /**
     * 新增管理员信息
     * @param {{adminname, adminpwd, adminemail, admintel, adminstatus}}
     * @return {Promise<boolean>} true代表新增成功,false代表新增失败
     */
    async add({adminname, adminpwd, adminemail, admintel, adminstatus}) {
        //现在在这里,要执行sql语句的新增之前,我们要把密码进行md5的加密
        adminpwd = md5( message: adminpwd+AppConfig.salt);
        let strSql = `INSERT INTO ${this.currentTableName} (adminname, adminpwd, adminemail, admintel, adminstatus) \
        let result = await this.executeSql(strSql, params: [adminname, adminpwd, adminemail, admintel, adminstatus]);
        return result.affectedRows > 0;
    }
```

在密码的后面就加了一个盐

完成上面的操作以后，当我们再去增管理员的进修，我们就可以看到，同样会新增成功，同样会得到一个加密的密码，如下所示

id	adminnam	adminpwd	adminemai	admintel	adminstatus	i
55	杨标	677bc50bda3009a6b3f452d57a4b389e	lovesnsfi@	18627193876	0	

但是当我们再去拿着这个加密的密码去解密的时候，就得到不出结果了

九、管理员的登录功能