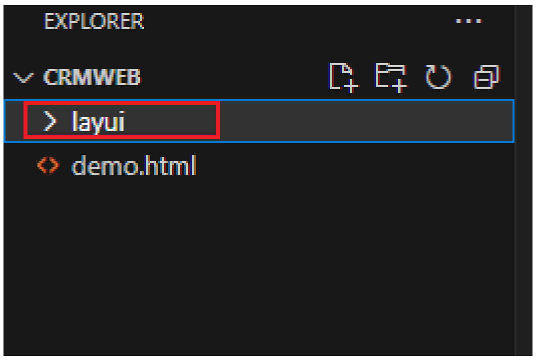
LayUI

引入LayUI

- 下载LayUI的包放到项目根目录下



- 在html页面中引入layui的css和js文件

```html
<!DOCTYPE html>
<html lang="en">
<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>Document</title>
    <link rel="stylesheet" href="./layui/css/layui.css">
</head>
<body>
    CRM前端
    <button type="button" class="layui-btn layui-btn-normal">百搭按钮</button>
    <div id="test1"></div>
<script src="./layui/layui.js"></script>
<script>
layui.use(['rate'], function(){
    var rate = layui.rate;

   rate.render({
    elem: '#test1'
  })
});
</script>
</body>
</html>
```

- 使用LayUI组件
  - 如果组件没有js代码，直接复制标签到页面，比如按钮

  ```html
  <button type="button" class="layui-btn layui-btn-normal">百搭按钮</button>
  ```

  - 如果组件还有js代码，需要标签和js代码，如：rate

标签：

```
<div id="test1"></div>
```

js代码：

```
<script>
//用到哪个组件模块，就要在layui.use方法里面导入模块，
layui.use(['rate'], function(){
    //为了使用组件的名称更短一些，通常会赋给一个变量
    var rate = layui.rate;

    //调用组件的render()方法（画出组件内容）
   rate.render({
    elem: '#test1'  //对应标签的id
  })
});
</script>
```

栅格布局

```
 <!-- 栅格布局 -->
    <div class="layui-fluid">
        <!-- 行 -->
        <div class="layui-row">
            <!-- 列，每行分为12份，每一列可以占多份 -->
            <div class="layui-col-md4">&nbsp;</div>
            <div class="layui-col-md4">中间放入文字内容</div>
            <div class="layui-col-md4">&nbsp;</div>
        </div>
    </div>
```

前端首页

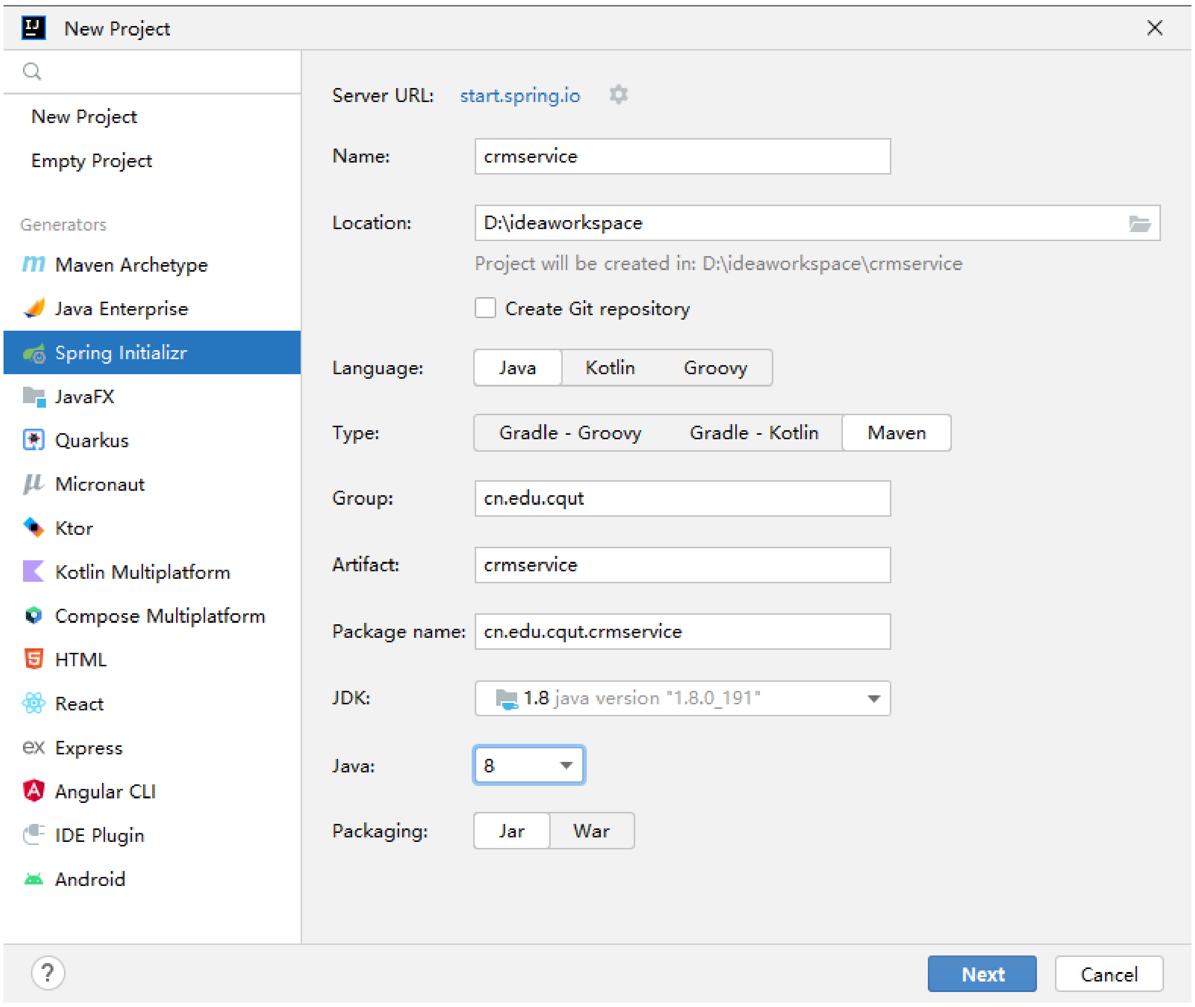
- 从官网复制后台管理布局页面
- 顶部菜单栏换成标题“CRM客户关系管理系统”
- 左上角的layui-log标签里面加图片
- 修改侧边栏的菜单
- 内容区域添加iframe，菜单点击的a标签加target指向iframe的name

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1, maximum-
scale=1">
  <title>layout 管理系统大布局 - Layui</title>
  <link rel="stylesheet" href="./layui/css/layui.css">
</head>
<body>
<div class="layui-layout layui-layout-admin">
```

```
  <div class="layui-header">
    <div class="layui-logo layui-hide-xs layui-bg-black">
        <img src="/img/logo.png" style="height: 40px;" alt="">
    </div>
    <!-- 头部区域（可配合layui 已有的水平导航） -->
    <ul class="layui-nav layui-layout-left">
      <li style="font-size: 24px;line-height: 60px;">CRM客户关系管理系统</li>
    </ul>
    <ul class="layui-nav layui-layout-right">
      <li class="layui-nav-item layui-hide layui-show-md-inline-block">
        <a href="javascript:;">
          <img
src="//tva1.sinaimg.cn/crop.0.0.118.118.180/5db11ff4gw1e77d3nqrv8j203b03cweg.jpg
" class="layui-nav-img">
          tester
        </a>
        <dl class="layui-nav-child">
          <dd><a href="">Your Profile</a></dd>
          <dd><a href="">Settings</a></dd>
          <dd><a href="">Sign out</a></dd>
        </dl>
      </li>
      <li class="layui-nav-item" lay-header-event="menuRight" lay-unselect>
        <a href="javascript:;">
          <i class="layui-icon layui-icon-more-vertical"></i>
        </a>
      </li>
    </ul>
  </div>

  <div class="layui-side layui-bg-black">
    <div class="layui-side-scroll">
      <!-- 左侧导航区域（可配合layui已有的垂直导航） -->
      <ul class="layui-nav layui-nav-tree" lay-filter="test">
        <li class="layui-nav-item layui-nav-itemed">
          <a class="" href="javascript:;">客户管理</a>
          <dl class="layui-nav-child">
            <!-- 页面在name叫content的iframe标签中显示 -->
            <dd><a href="customerlist.html" target="content">客户信息</a></dd>
            <dd><a href="demo.html" target="content">客户联系人</a></dd>
            <dd><a href="javascript:;">客户交往记录</a></dd>

          </dl>
        </li>
        <li class="layui-nav-item">
          <a href="javascript:;">营销管理</a>
          <dl class="layui-nav-child">
            <dd><a href="javascript:;">list 1</a></dd>
            <dd><a href="javascript:;">list 2</a></dd>
            <dd><a href="">超链接</a></dd>
          </dl>
        </li>
        <li class="layui-nav-item">
            <a href="javascript:;">服务管理</a>
            <dl class="layui-nav-child">
              <dd><a href="javascript:;">list 1</a></dd>
              <dd><a href="javascript:;">list 2</a></dd>
              <dd><a href="">超链接</a></dd>
            </dl>
          </li>
          <li class="layui-nav-item">
            <a href="javascript:;">统计报表</a>
            <dl class="layui-nav-child">
              <dd><a href="javascript:;">list 1</a></dd>
              <dd><a href="javascript:;">list 2</a></dd>
              <dd><a href="">超链接</a></dd>
            </dl>
          </li>
      </ul>
    </div>
  </div>

  <div class="layui-body">
    <!-- 内容主体区域 -->
    <div style="padding: 15px;">
        <iframe src="" name="content" frameborder="0"></iframe>
    </div>
  </div>

  <div class="layui-footer">
    <!-- 底部固定区域 -->
    底部固定区域
  </div>
</div>
<script src="./layui/layui.js"></script>
<script>
//JS
layui.use(['element', 'layer', 'util'], function(){
  var element = layui.element
  ,layer = layui.layer
  ,util = layui.util
  ,$ = layui.$;

  //头部事件
  util.event('lay-header-event', {
    //左侧菜单事件
    menuLeft: function(othis){
      layer.msg('展开左侧菜单的操作', {icon: 0});
    }
    ,menuRight: function(){
      layer.open({
        type: 1
        ,content: '<div style="padding: 15px;">处理右侧面板的操作</div>'
        ,area: ['260px', '100%']
        ,offset: 'rt' //右上角
        ,anim: 5
        ,shadeClose: true
      });
    }
  });

});
</script>
</body>
</html>
```

创建Spring Boot项目

- 创建新项目，选择Spring Innitializr



Server URL: start.spring.io

Name: crmservice

Location: D:\ideaworkspace

Project will be created in: D:\ideaworkspace\crmservice

Create Git repository

Language: [✓ Java] Kotlin Groovy

Type: Gradle - Groovy Gradle - Kotlin [✓ Maven]

Group: cn.edu.cqut

Artifact: crmservice

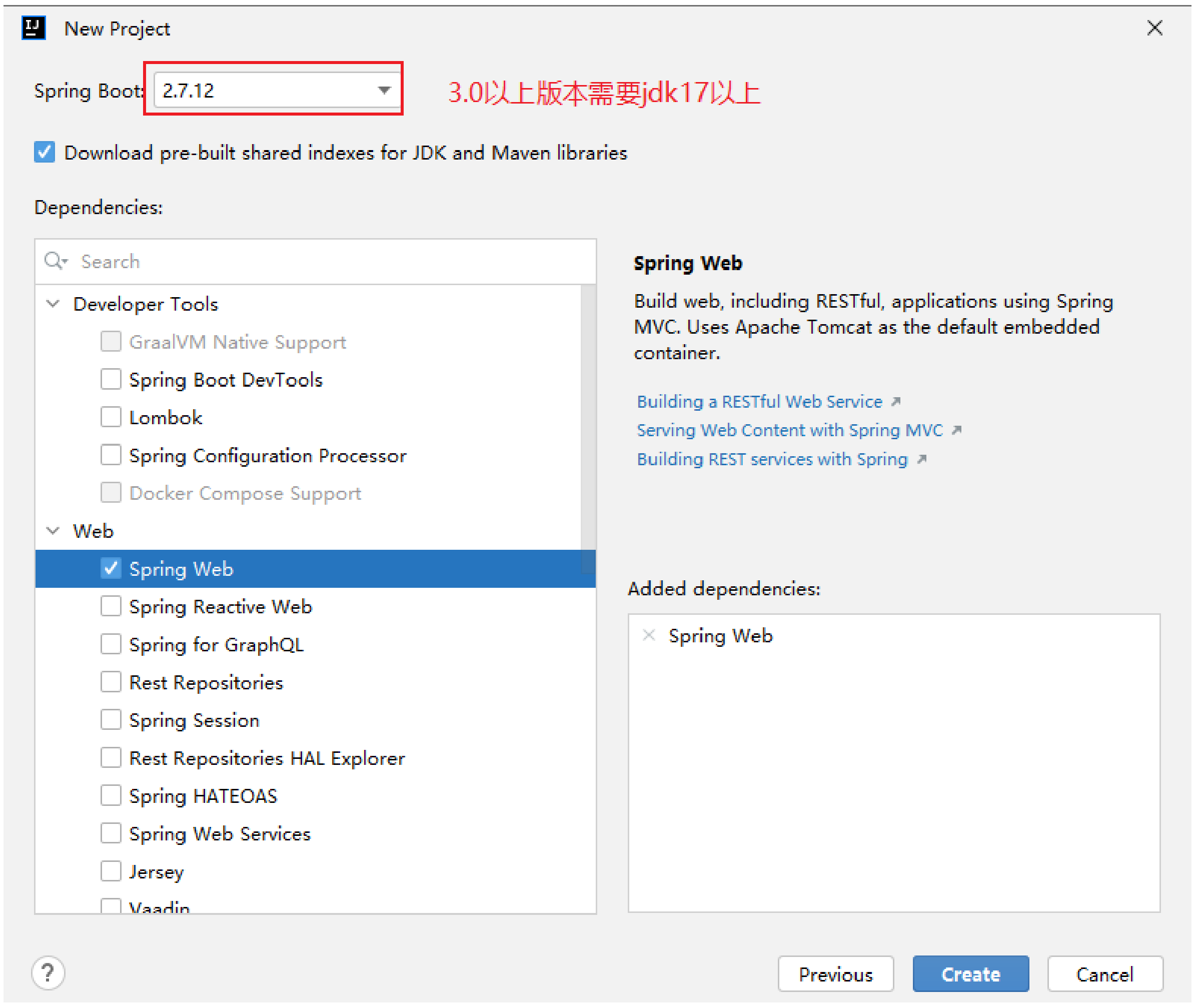
Package name: cn.edu.cqut.crmservice

JDK: 1.8 java version "1.8.0_191"

Java: 8

Packaging: [✓ Jar] War

- 选择SpringBoot版本，添加依赖包


New Project

Spring Boot: 2.7.12

3.0以上版本需要jdk17以上

☑ Download pre-built shared indexes for JDK and Maven libraries

Dependencies:

- Developer Tools
  - ☐ GraalVM Native Support
  - ☐ Spring Boot DevTools
  - ☐ Lombok
  - ☐ Spring Configuration Processor
  - ☐ Docker Compose Support
- Web
  - ☑ Spring Web
  - ☐ Spring Reactive Web
  - ☐ Spring for GraphQL
  - ☐ Rest Repositories
  - ☐ Spring Session
  - ☐ Rest Repositories HAL Explorer
  - ☐ Spring HATEOAS
  - ☐ Spring Web Services
  - ☐ Jersey
  - ☐ Vaadin

Spring Web

Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Building a RESTful Web Service ↗
Serving Web Content with Spring MVC ↗
Building REST services with Spring ↗

Added dependencies:

Spring Web

添加MyBatis Plus支持

网址： https://baomidou.com/

添加包依赖

在pom.xml文件中增加如下内容, 然后点击maven的reload按钮下载包

```xml
<!--        mybatis plus 数据库访问框架-->
        <dependency>
            <groupId>com.baomidou</groupId>
            <artifactId>mybatis-plus-boot-starter</artifactId>
            <version>3.5.3</version>
        </dependency>
<!--        mybatis plus 根据数据库逆向生成代码工具-->
        <dependency>
            <groupId>com.baomidou</groupId>
            <artifactId>mybatis-plus-generator</artifactId>
            <version>3.5.3</version>
        </dependency>
        <dependency>
            <groupId>org.apache.velocity</groupId>
            <artifactId>velocity-engine-core</artifactId>
            <version>2.3</version>
        </dependency>
```

```
<!--        mysql jdbc 驱动-->
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
        </dependency>
```

配置数据库的JDBC连接

在application.properties文件中添加配置

```
#数据库链接信息
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/crmdb?
serverTimezone=Asia/Shanghai
spring.datasource.username=root
spring.datasource.password=123456
```

根据表逆向生成代码

创建一个生成代码的工具类，然后运行

```
package cn.edu.cqut.crmservice.util;

import com.baomidou.mybatisplus.generator.FastAutoGenerator;

public class Generator {
    public static void main(String[] args) {
        FastAutoGenerator.create("jdbc:mysql://localhost:3306/crmdb?
serverTimezone=Asia/Shanghai", "root", "123456")
                .globalConfig(builder -> {
                    builder.author("CQUT") // 设置作者

.outputDir("D:\\ideaworkspace\\crmservice\\src\\main\\java"); // 指定输出目录
                })
                .packageConfig(builder -> {
                    builder.parent("cn.edu.cqut.crmservice"); // 设置父包名
                })
                .strategyConfig(builder -> {
                    builder.addInclude("customer"); // 设置需要生成的表名
                })
                .execute();
    }
}
```

客户信息管理

客户信息列表

- 后台CustomerController类增加一个查询方法

```
package cn.edu.cqut.crmservice.controller;
```

```java
import cn.edu.cqut.crmservice.entity.Customer;
import cn.edu.cqut.crmservice.service.ICustomerService;
import cn.edu.cqut.crmservice.util.TableResult;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RestController;

import java.util.List;

/**
 * <p>
 *   前端控制器
 * </p>
 *
 * @author CQUT
 * @since 2023-06-06
 */
@RestController  //给前端返回json数据
@RequestMapping("/customer")
@CrossOrigin //允许跨域请求
public class CustomerController {
    @Autowired  //自动从spring容器中获取对象给变量赋值
    private ICustomerService customerService;

    @GetMapping("/getCustomerList")
    public TableResult getCustomerList(){
        List<Customer> list = customerService.list();//调用service层的list方法，返回
数据表的所有数据
        TableResult result = new TableResult();
        result.setCode(0);  //后台返回成功
        result.setCount(list.size());
        result.setMsg("后台查询成功");  //数据表格遇到异常时显示的提示文字
        result.setData(list);
        return result;
    }
}
```

- 为了匹配前端数据表格需要的json格式，创建一个返回类型TableResult

```java
package cn.edu.cqut.crmservice.util;

import cn.edu.cqut.crmservice.entity.Customer;

import java.util.List;

public class TableResult {
    private int code;
    private String msg;
    private long count;
    private List<Customer> data;

    public int getCode() {
        return code;
    }

    public void setCode(int code) {
        this.code = code;
    }

    public String getMsg() {
        return msg;
    }

    public void setMsg(String msg) {
        this.msg = msg;
    }

    public long getCount() {
        return count;
    }

    public void setCount(long count) {
        this.count = count;
    }

    public List<Customer> getData() {
        return data;
    }

    public void setData(List<Customer> data) {
        this.data = data;
    }
}
```

- 前端创建页面customerlist.html

创建一个数据表格，表格url是后台方法映射的url地址，表格的列设置为跟Customer属性相同

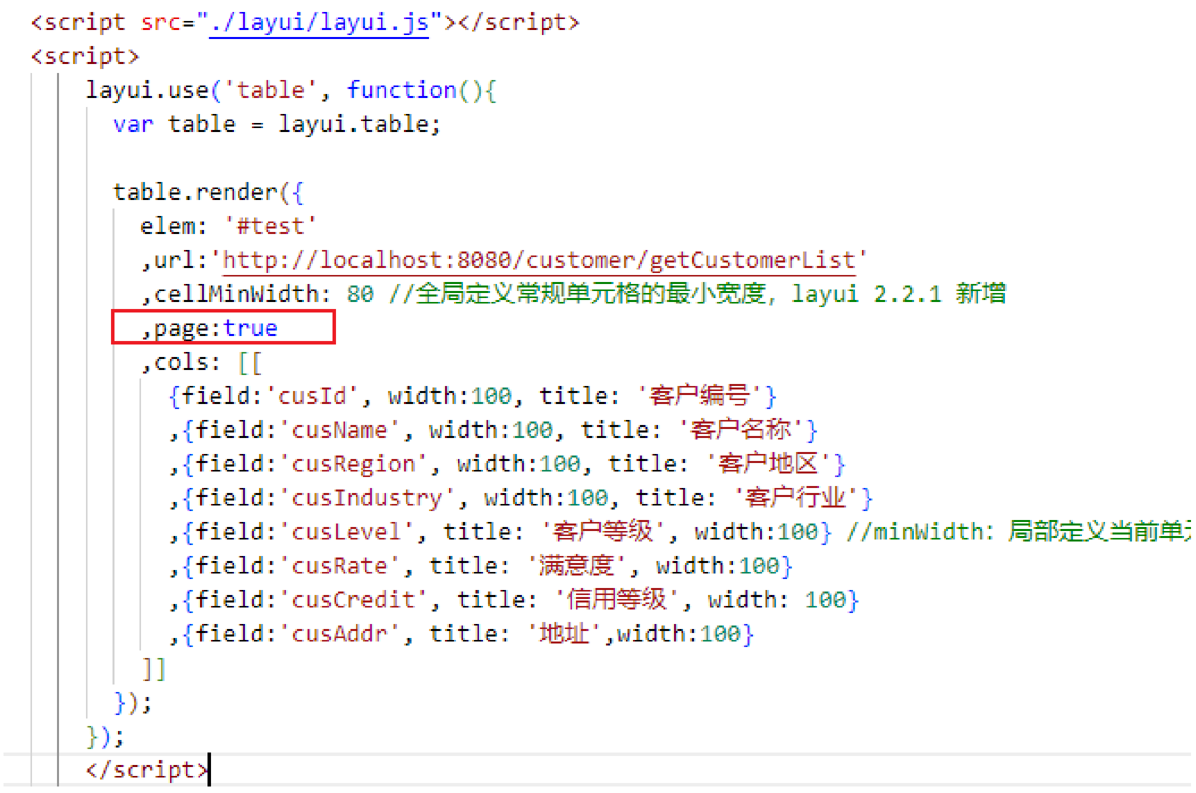
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <link rel="stylesheet" href="./layui/css/layui.css">
</head>
<body>
    <table class="layui-hide" id="test"></table>

    <script src="./layui/layui.js"></script>
    <script>
        layui.use('table', function(){
          var table = layui.table;

          table.render({
            elem: '#test'
            ,url:'http://localhost:8080/customer/getCustomerList'
```

```
              ,cellMinWidth: 80 //全局定义常规单元格的最小宽度, layui 2.2.1 新增
              ,cols: [[
                {field:'cusId', width:100, title: '客户编号'}
                ,{field:'cusName', width:100, title: '客户名称'}
                ,{field:'cusRegion', width:100, title: '客户地区'}
                ,{field:'cusIndustry', width:100, title: '客户行业'}
                ,{field:'cusLevel', title: '客户等级', width:100} //minWidth: 局部定义
当前单元格的最小宽度, layui 2.2.1 新增
                ,{field:'cusRate', title: '满意度', width:100}
                ,{field:'cusCredit', title: '信用等级', width: 100}
                ,{field:'cusAddr', title: '地址',width:100}
              ]]
            });
          });
        </script>
    </body>
</html>
```

分页查询

- 前端的数据表格允许分页

table的render（）方法的参数添加 page:true



```
<script src="./layui/layui.js"></script>
<script>
    layui.use('table', function(){
      var table = layui.table;

      table.render({
        elem: '#test'
        ,url:'http://localhost:8080/customer/getCustomerList'
        ,cellMinWidth: 80 //全局定义常规单元格的最小宽度, layui 2.2.1 新增
        ,page:true
        ,cols: [[
          {field:'cusId', width:100, title: '客户编号'}
          ,{field:'cusName', width:100, title: '客户名称'}
          ,{field:'cusRegion', width:100, title: '客户地区'}
          ,{field:'cusIndustry', width:100, title: '客户行业'}
          ,{field:'cusLevel', title: '客户等级', width:100} //minWidth: 局部定义当前单
          ,{field:'cusRate', title: '满意度', width:100}
          ,{field:'cusCredit', title: '信用等级', width: 100}
          ,{field:'cusAddr', title: '地址',width:100}
        ]]
      });
    });
    </script>
```

- 配置分页插件

在启动类(CrmserviceApplication)中添加一个方法如下：

```
    //分页插件的配置
    @Bean
    public MybatisPlusInterceptor mybatisPlusInterceptor(){
        MybatisPlusInterceptor interceptor = new MybatisPlusInterceptor();
        interceptor.addInnerInterceptor(new PaginationInnerInterceptor());
        return interceptor;
    }
```

- 修改后台的查询方法

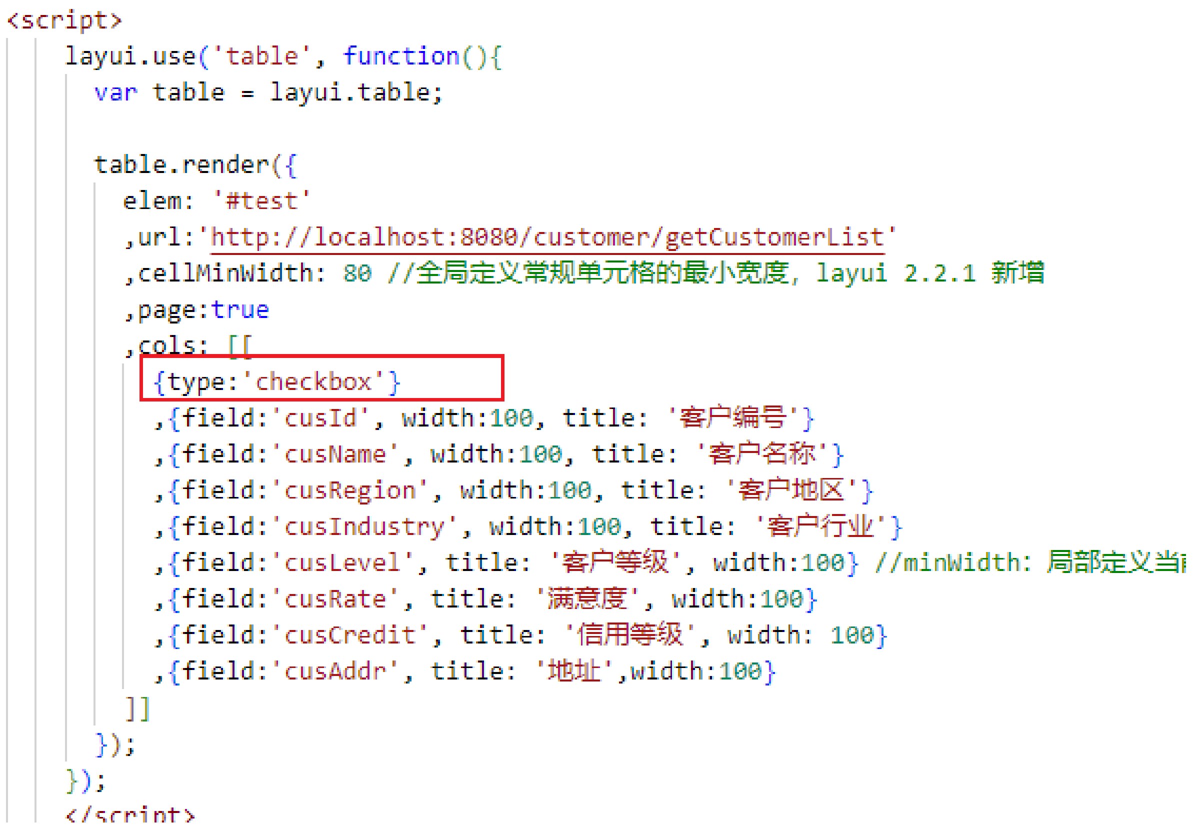
```
 /**
     *
     * @param limit 每页行数
     * @param page 第几页
     * @return
     */
    @GetMapping("/getCustomerList")
    public TableResult<Customer> getCustomerList(Integer limit, Integer page){
        Page<Customer> customerPage = new Page<>(page, limit);
        Page<Customer> page1 = customerService.page(customerPage);//调用service层的page方法，返回分页
        //getTotal()方法返回表里面的总记录数，    getRecords()方法返回当前页的数据列表
        return TableResult.ok("查询成功", page1.getTotal(), page1.getRecords());
    }
```

修改

前端页面

- 首先让表格可以选择行

在第一列添加复选框



```
<script>
  layui.use('table', function(){
    var table = layui.table;

    table.render({
      elem: '#test'
      ,url:'http://localhost:8080/customer/getCustomerList'
      ,cellMinWidth: 80 //全局定义常规单元格的最小宽度，layui 2.2.1 新增
      ,page:true
      ,cols: [[
        {type:'checkbox'}
        ,{field:'cusId', width:100, title: '客户编号'}
        ,{field:'cusName', width:100, title: '客户名称'}
        ,{field:'cusRegion', width:100, title: '客户地区'}
        ,{field:'cusIndustry', width:100, title: '客户行业'}
        ,{field:'cusLevel', title: '客户等级', width:100} //minWidth: 局部定义当
        ,{field:'cusRate', title: '满意度', width:100}
        ,{field:'cusCredit', title: '信用等级', width: 100}
        ,{field:'cusAddr', title: '地址',width:100}
      ]]
    });
  });
</script>
```

- 添加表头工具栏

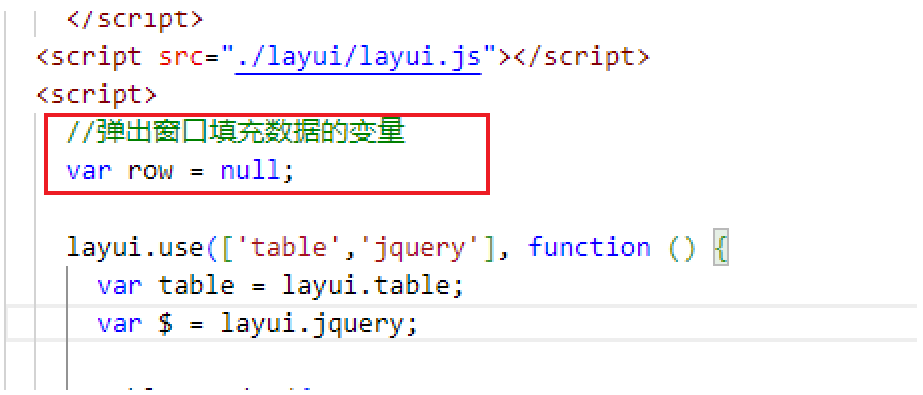
```
<script type="text/html" id="toolbarDemo">
      <div class="layui-btn-container">
        <button class="layui-btn layui-btn-sm" lay-event="edit">修改</button>
      </div>
    </script>
```

- 表头工具栏事件中弹出窗口

```
//头工具栏事件
      table.on('toolbar(test)', function (obj) {
        var checkStatus = table.checkStatus(obj.config.id);
        switch (obj.event) {
          case 'edit':
            //选中行的数据的数组
            var data = checkStatus.data;
            if (data.length == 0) {
              layer.msg("请选择要修改的行")
            } else if (data.length > 1) {
              layer.msg("一次只能修改一行数据")
            } else {
              row = data[0]  //把选中行对象赋给row变量
              //弹出窗口
              layer.open({
                type: 2 //此处以iframe举例
                , title: '修改客户信息'
                , area: ['390px', '450px']
                , shade: 0.3   //背景透明度，取值范围0~1
                , maxmin: true  //窗口是否允许最大化和最小化
                , offset: [ //居中显示
                  ($(window).height() - 450)/2
                  , ($(window).width() - 390)/2
                ]
                , content: 'customerupdate.html' //弹出窗口的页面内容
              });
            }
            break;
        };
      });
```

- 列表页面还要定义全局变量用于弹出窗口表单赋值

  定义在script之后，layui.use()方法前



```
  </script>
<script src="./layui/layui.js"></script>
<script>
  //弹出窗口填充数据的变量
  var row = null;

  layui.use(['table','jquery'], function () {
    var table = layui.table;
    var $ = layui.jquery;
```

完整的customerlist.html

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

<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>Document</title>
  <link rel="stylesheet" href="./layui/css/layui.css">
</head>

<body>
  <table class="layui-hide" id="test" lay-filter="test"></table>
  <script type="text/html" id="toolbarDemo">
      <div class="layui-btn-container">
        <button class="layui-btn layui-btn-sm" lay-event="edit">修改</button>
      </div>
    </script>
  <script src="./layui/layui.js"></script>
  <script>
    //弹出窗口填充数据的变量
    var row = null;

    layui.use(['table','jquery'], function () {
      var table = layui.table;
      var $ = layui.jquery;

      table.render({
        elem: '#test'
        , url: 'http://localhost:8080/customer/getCustomerList'
        , cellMinWidth: 80 //全局定义常规单元格的最小宽度，layui 2.2.1 新增
        , toolbar: '#toolbarDemo' //开启头部工具栏，并为其绑定左侧模板
        , page: true
        , cols: [[
          { type: 'checkbox' }
          , { field: 'cusId', width: 100, title: '客户编号' }
          , { field: 'cusName', width: 100, title: '客户名称' }
          , { field: 'cusRegion', width: 100, title: '客户地区' }
          , { field: 'cusIndustry', width: 100, title: '客户行业' }
          , { field: 'cusLevel', title: '客户等级', width: 100 } //minWidth：局部定
义当前单元格的最小宽度，layui 2.2.1 新增
          , { field: 'cusRate', title: '满意度', width: 100 }
          , { field: 'cusCredit', title: '信用等级', width: 100 }
          , { field: 'cusAddr', title: '地址', width: 100 }
        ]]
      });

      //头工具栏事件
      table.on('toolbar(test)', function (obj) {
        var checkStatus = table.checkStatus(obj.config.id);
        switch (obj.event) {
          case 'edit':
            //选中行的数据的数组
            var data = checkStatus.data;
            if (data.length == 0) {
              layer.msg("请选择要修改的行")
            } else if (data.length > 1) {
              layer.msg("一次只能修改一行数据")
            } else {
              row = data[0]  //把选中行对象赋给row变量
              //弹出窗口
              layer.open({
                type: 2 //此处以iframe举例
                , title: '修改客户信息'
                , area: ['390px', '450px']
                , shade: 0.3   //背景透明度，取值范围0~1
                , maxmin: true  //窗口是否允许最大化和最小化
                , offset: [ //居中显示
                  ($(window).height() - 450)/2
                  , ($(window).width() - 390)/2
                ]
                , content: 'customerupdate.html' //弹出窗口的页面内容
              });
            }
            break;
        };
      });

    });
  </script>
</body>

</html>
```

- 修改页面customerupdate.html

完成修改表单，标签的name与实体的属性保持一致

把选择行的数据填充到表单

写ajax提交的方法，提交成功后刷新表格，关闭窗口

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

<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>Document</title>
    <link rel="stylesheet" href="./layui/css/layui.css">
</head>

<body style="padding: 10px;">

    <form class="layui-form" lay-filter="updateCustomerForm">
        <!-- 用隐藏的输入框保存主键（客户编号），后台做更新时候要根据主键做update -->
        <input type="hidden" name="cusId" />
        <div class="layui-form-item"> <label class="layui-form-label">客户名称
</label>
            <div class="layui-input-block">
                <input type="text" name="cusName" autocomplete="off"
placeholder="请输入客户名称" class="layui-input">
            </div>
        </div>

        <div class="layui-form-item"> <label class="layui-form-label">客户地区
</label>
            <div class="layui-input-block">
                <select name="cusRegion">
                    <option value=""></option>
                    <option value="东北">东北</option>
                    <option value="西北">西北</option>
                    <option value="华北">华北</option>
                    <option value="华中">华中</option>
                    <option value="华南">华南</option>
                    <option value="西南">西南</option>
                    <option value="华东">华东</option>
                </select>
            </div>
        </div>

        <div class="layui-form-item"> <label class="layui-form-label">客户行业
</label>
            <div class="layui-input-block">
                <select name="cusIndustry">
                    <option value=""></option>
                    <option value="教育">教育</option>
                    <option value="医疗">医疗</option>
                    <option value="金融">金融</option>
                    <option value="制造">制造</option>
                    <option value="服务">服务</option>
                </select>
            </div>
        </div>
        <div class="layui-form-item"> <label class="layui-form-label">客户等级
</label>
            <div class="layui-input-block">
                <select name="cusLevel">
                    <option value=""></option>
                    <option value="VIP客户">VIP客户</option>
                    <option value="大客户">大客户</option>
                    <option value="普通客户">普通客户</option>
                </select>
            </div>
        </div>

        <div class="layui-form-item"> <label class="layui-form-label">客户满意度
</label>
            <div class="layui-input-block">
                <select name="cusRate">
                    <option value=""></option>
                    <option value="1">一星</option>
                    <option value="2">二星</option>
                    <option value="3">三星</option>
                    <option value="4">四星</option>
                    <option value="5">五星</option>
                </select>
            </div>
        </div>

        <div class="layui-form-item"> <label class="layui-form-label">客户信用度
</label>
            <div class="layui-input-block">
                <select name="cusCredit">
                    <option value=""></option>
                    <option value="1">一星</option>
                    <option value="2">二星</option>
                    <option value="3">三星</option>
                    <option value="4">四星</option>
                    <option value="5">五星</option>
                </select>
            </div>
        </div>

        <div class="layui-form-item"> <label class="layui-form-label">客户地址
</label>
            <div class="layui-input-block">
                <input type="text" name="cusAddr" autocomplete="off"
placeholder="请输入客户地址" class="layui-input">
            </div>
        </div>

        <div class="layui-form-item"> <label class="layui-form-label">客户邮编
</label>
            <div class="layui-input-block">
                <input type="text" name="cusPostcode" autocomplete="off"
placeholder="请输入客户邮编" class="layui-input">
            </div>
        </div>

        <div class="layui-form-item"> <label class="layui-form-label">客户电话
</label>
```

```
            <div class="layui-input-block">
                <input type="text" name="cusPhone" autocomplete="off"
placeholder="请输入客户电话" class="layui-input">
            </div>
        </div>

        <div class="layui-form-item"> <label class="layui-form-label">客户网址
</label>
            <div class="layui-input-block">
                <input type="text" name="cusUrl" autocomplete="off"
placeholder="请输入客户网址" class="layui-input">
            </div>
        </div>

        <div class="layui-form-item"> <label class="layui-form-label">客户传真
</label>
            <div class="layui-input-block">
                <input type="text" name="cusFax" autocomplete="off"
placeholder="请输入客户传真" class="layui-input">
            </div>
        </div>


        <div class="layui-form-item">
            <div class="layui-input-block">
                <button type="submit" class="layui-btn" lay-submit="" lay-
filter="submitCustomerUpdate">立即提交</button>
                <button type="reset" class="layui-btn layui-btn-primary">重置
</button>
            </div>
        </div>
    </form>


    <script src="./layui/layui.js"></script>
    <script>
        layui.use(['form', 'jquery'], function () {
            var form = layui.form;
            var $ = layui.jquery;

            //给表单填充内容
            form.val('updateCustomerForm', parent.row);

            //监听提交
            form.on('submit(submitCustomerUpdate)', function (data) {
                //异步（ajax）提交代码
                $.ajax({
                    type: "post",
                    url:"http://localhost:8080/customer/updateCustomer",
                    dataType:"json",
                    data: data.field,
                    success: function(obj){ //后台方法成功执行并返回结果时，会调用这个方
法，参数是后台返回的内容
                        //刷新表格
                        parent.table.reload('test', {})

                        //关闭窗口
                        var index = parent.layer.getFrameIndex(window.name); //先
得到当前iframe层的索引
                        parent.layer.close(index); //再执行关闭
                    }
                })

                //避免页面因form提交而刷新
                return false;
            });


        })
    </script>
</body>

</html>
```

后台

在CustomerController类增加一个修改的方法

```
 @PostMapping("/updateCustomer")
    public TableResult<Customer> updateCustomer(Customer customer){
        customerService.updateById(customer);
        return TableResult.ok("修改客户信息成功");
    }
```

新增

前端页面

从修改页面复制一份，然后改为新增页面， 填充表单数据功能不需要，主键的输入框也不需要

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

<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>Document</title>
    <link rel="stylesheet" href="./layui/css/layui.css">
</head>

<body style="padding: 10px;">

    <form class="layui-form" lay-filter="addCustomerForm">
        <div class="layui-form-item"> <label class="layui-form-label">客户名称
</label>
            <div class="layui-input-block">
                <input type="text" name="cusName" autocomplete="off"
placeholder="请输入客户名称" class="layui-input">
            </div>
        </div>
        <div class="layui-form-item"> <label class="layui-form-label">客户地区
</label>
            <div class="layui-input-block">
                <select name="cusRegion">
                    <option value=""></option>
                    <option value="东北">东北</option>
                    <option value="西北">西北</option>
                    <option value="华北">华北</option>
                    <option value="华中">华中</option>
                    <option value="华南">华南</option>
                    <option value="西南">西南</option>
                    <option value="华东">华东</option>
                </select>
            </div>
        </div>

        <div class="layui-form-item"> <label class="layui-form-label">客户行业
</label>
            <div class="layui-input-block">
                <select name="cusIndustry">
                    <option value=""></option>
                    <option value="教育">教育</option>
                    <option value="医疗">医疗</option>
                    <option value="金融">金融</option>
                    <option value="制造">制造</option>
                    <option value="服务">服务</option>
                </select>
            </div>
        </div>

        <div class="layui-form-item"> <label class="layui-form-label">客户等级
</label>
            <div class="layui-input-block">
                <select name="cusLevel">
                    <option value=""></option>
                    <option value="VIP客户">VIP客户</option>
                    <option value="大客户">大客户</option>
                    <option value="普通客户">普通客户</option>
                </select>
            </div>
        </div>

        <div class="layui-form-item"> <label class="layui-form-label">客户满意度
</label>
            <div class="layui-input-block">
                <select name="cusRate">
                    <option value=""></option>
                    <option value="1">一星</option>
                    <option value="2">二星</option>
                    <option value="3">三星</option>
                    <option value="4">四星</option>
                    <option value="5">五星</option>
                </select>
            </div>
        </div>

        <div class="layui-form-item"> <label class="layui-form-label">客户信用度
</label>
```

```
            <div class="layui-input-block">
                <select name="cusCredit">
                    <option value=""></option>
                    <option value="1">一星</option>
                    <option value="2">二星</option>
                    <option value="3">三星</option>
                    <option value="4">四星</option>
                    <option value="5">五星</option>
                </select>
            </div>
        </div>

        <div class="layui-form-item"> <label class="layui-form-label">客户地址
</label>
            <div class="layui-input-block">
                <input type="text" name="cusAddr" autocomplete="off"
placeholder="请输入客户地址" class="layui-input">
            </div>
        </div>

        <div class="layui-form-item"> <label class="layui-form-label">客户邮编
</label>
            <div class="layui-input-block">
                <input type="text" name="cusPostcode" autocomplete="off"
placeholder="请输入客户邮编" class="layui-input">
            </div>
        </div>

        <div class="layui-form-item"> <label class="layui-form-label">客户电话
</label>
            <div class="layui-input-block">
                <input type="text" name="cusPhone" autocomplete="off"
placeholder="请输入客户电话" class="layui-input">
            </div>
        </div>

        <div class="layui-form-item"> <label class="layui-form-label">客户网址
</label>
            <div class="layui-input-block">
                <input type="text" name="cusUrl" autocomplete="off"
placeholder="请输入客户网址" class="layui-input">
            </div>
        </div>

        <div class="layui-form-item"> <label class="layui-form-label">客户传真
</label>
            <div class="layui-input-block">
                <input type="text" name="cusFax" autocomplete="off"
placeholder="请输入客户传真" class="layui-input">
            </div>
        </div>



        <div class="layui-form-item">
            <div class="layui-input-block">
                <button type="submit" class="layui-btn" lay-submit="" lay-
filter="submitCustomerAdd">立即提交</button>
```

```
                <button type="reset" class="layui-btn layui-btn-primary">重置
</button>
            </div>
        </div>
    </form>


    <script src="./layui/layui.js"></script>
    <script>
        layui.use(['form', 'jquery'], function () {
            var form = layui.form;
            var $ = layui.jquery;

            //监听提交
            form.on('submit(submitCustomerAdd)', function (data) {
                //异步（ajax）提交代码
                $.ajax({
                    type: "post",
                    url:"http://localhost:8080/customer/addCustomer",
                    dataType:"json",
                    data: data.field,
                    success: function(obj){ //后台方法成功执行并返回结果时，会调用这个方
法，参数是后台返回的内容
                        //刷新表格
                        parent.table.reload('test', {})

                        //关闭窗口
                        var index = parent.layer.getFrameIndex(window.name); //先
得到当前iframe层的索引
                        parent.layer.close(index); //再执行关闭
                    }
                })

                //避免页面因form提交而刷新
                return false;
            });


        })
    </script>
</body>

</html>
```

后台

在CustomerController类增加一个新增的方法

```
    @PostMapping("/addCustomer")
    public TableResult<Customer> addCustomer(Customer customer){
        customerService.save(customer);
        return TableResult.ok("新增客户信息成功");
    }
```

删除

- 前端

```
case 'delete': $
            if (data.length < 1) {
              layer.msg("请选择要删除的行")
            } else {
              layer.confirm('确认要删除选中的行吗?', function (index) {
                //把选中行的cusId保存到数组
                var arr = [];
                for(var i=0; i<data.length; i++){
                  var cusId = data[i].cusId;
                  arr.push(cusId);
                }

                $.ajax({
                    type: "post",
                    url:"http://localhost:8080/customer/deleteCustomer",
                    dataType:"json",
                    data: {
                      ids: arr.join(",")    //用数组的join方法把数组元素用,拼接成一个字
符串，结果如：1,3,5
                    },
                    success: function(obj){ //后台方法成功执行并返回结果时，会调用这个方
法，参数是后台返回的内容
                        //刷新表格
                        table.reload('test', {})
                    }
                })
                layer.close(index);
              });
            }
            break;
```

- 后台

```
@PostMapping("/deleteCustomer")
    public TableResult<Customer> deleteCustomer(Integer[] ids){//参数名要跟前端ajax
方法data参数里面的属性名一致
        customerService.removeByIds(Arrays.asList(ids)); //asList()把数组转list
        return TableResult.ok("删除客户信息成功");
    }
```

数据库中存状态码，显示转换方案

有时候数据库中存入的是数字或状态，但显示时要显示文字，比如性别，数据库存的是1或0，但显示时要显示男或女

方案一：后台转换

在实体类中增加一个用于显示的属性，在该属性的get方法中进行转换，该属性必须要添加注解：

```
@TableField(exist = false) //属性在数据库里面没有对应字段
```

在前端显示时就用这个新增的属性

方案二： 前端转换

前端转换的方式不同框架可能略有不同，在layui的table中是设置表头参数templet来实现

参数d就是当前行的数据对象，函数返回的值会显示在当列

```
 { field: 'cusRate', title: '满意度', width: 100, templet:function(d){
            var ret = "";
            switch(d.cusRate){
              case 1:
                 ret = "一星"
              break;
              case 2:
                ret = "二星"
              break;
              case 3:
                ret = "三星"
              break;
              case 4:
                ret = "四星"
              break;
              case 5:
                ret = "五星"
              break;
            }
            return ret;
          } }
```

MyBatis Plus条件查询

在service的查询方法中添加QueryWrapper参数， 可以给QueryWrapper对象设置多个查询条件，默认每个条件之间用and 连接， 如果要用or连接，在设置条件之前先执行or（）方法

```
 @GetMapping("/getContactPage")
    public TableResult<Contact> getContactPage(Integer page, Integer limit,
Contact contact){
        //条件查询
        QueryWrapper<Contact> wrapper = new QueryWrapper<>();
        wrapper.eq("cus_id", contact.getCusId()); //第一个参数是字段名

        Page<Contact> page1 = new Page<>(page, limit);
        Page<Contact> contactPage = contactService.page(page1, wrapper);
        return TableResult.ok("查询客户联系人成功", contactPage.getTotal(),
contactPage.getRecords());
    }
```

前端页面之间跳转及参数传递

- 在js中跳转页面

```
window.location.href="contactlist.html?cusId=" + data[0].cusId
```

- 在前端页面接受url参数

```
/获取客户编号的参数
      var url = window.location.href;
//http://127.0.0.1:5500/contactlist.html?cusId=1
      var arr = url.split("?");
      var param = "";
      if(arr.length > 1){
        param = "?" + arr[1]
      }
```

用户和权限管理

创建用户表sys_user

```
CREATE TABLE `sys_user`  (
  `su_id` int(0) NOT NULL AUTO_INCREMENT COMMENT '用户编号，主键，自动递增',
  `su_name` varchar(50) CHARACTER SET utf8 COLLATE utf8_general_ci NOT NULL
COMMENT '用户名',
  `su_pwd` varchar(50) CHARACTER SET utf8 COLLATE utf8_general_ci NOT NULL
COMMENT '用户密码',
  `su_role` varchar(50) CHARACTER SET utf8 COLLATE utf8_general_ci NOT NULL
COMMENT '用户角色',
  PRIMARY KEY (`su_id`) USING BTREE
) ENGINE = InnoDB AUTO_INCREMENT = 4 CHARACTER SET = utf8 COLLATE =
utf8_general_ci ROW_FORMAT = Dynamic;

-- ----------------------------
-- Records of sys_user
-- ----------------------------
INSERT INTO `sys_user` VALUES (1, 'admin', '123456', 'ADMIN');
INSERT INTO `sys_user` VALUES (2, 'user1', '123456', 'SALES');
INSERT INTO `sys_user` VALUES (3, 'user2', '123456', 'MANAGER');
```

生成sys_user表的相关类文件

pom.xml文件中添加依赖，然后刷新maven

```
        <!-- jwt 权限管理 -->
        <dependency>
            <groupId>io.jsonwebtoken</groupId>
            <artifactId>jjwt</artifactId>
            <version>0.9.1</version>
        </dependency>
        <!-- 用于java对象和JSON字符串互转 -->
        <dependency>
            <groupId>com.alibaba</groupId>
            <artifactId>fastjson</artifactId>
            <version>1.2.75</version>
        </dependency>
```

修改一下返回前端的类

添加异常的方法

```java

import cn.edu.cqut.crmservice.entity.Customer;

import java.util.List;

public class TableResult<T> {
    //    后台返回的状态码，0-成功；  其它值不成功
    private int code;
    // 后台返回的提示信息。如果请求失败时，数据表格会把提示信息显示出来
    private String msg;
    //表里的总记录数，用于计算分页
    private long count;
    //当前页显示的数据
    private List<T> data;
    //实体对象
    private T obj;

    public TableResult(int code, String msg, long count, List<T> data, T obj) {
        this.code = code;
        this.msg = msg;
        this.count = count;
        this.data = data;
        this.obj = obj;
    }

    public static <T> TableResult<T> ok(String msg, long count, List<T> data){
        return new TableResult<T>(0, msg, count, data, null);
    }

    public static <T> TableResult<T> ok(String msg){
        return new TableResult<T>(0, msg, 0, null, null);
    }

    public static <T> TableResult<T> ok(String msg, T obj){
        return new TableResult<T>(0, msg, 0, null, obj);
    }
    public static <T> TableResult<T> error(int code, String msg){
        return new TableResult<T>(1, msg, code, null, null);
    }


    public int getCode() {
        return code;
    }

    public void setCode(int code) {
        this.code = code;
    }

    public String getMsg() {
        return msg;
    }

    public void setMsg(String msg) {
        this.msg = msg;
    }

    public long getCount() {
        return count;
    }

    public void setCount(long count) {
        this.count = count;
    }

    public List<T> getData() {
        return data;
    }

    public void setData(List<T> data) {
        this.data = data;
    }

    public T getObj() {
        return obj;
    }

    public void setObj(T obj) {
        this.obj = obj;
    }
}
```

创建JWT工具类

用于生成token, 校验token

```
import cn.edu.cqut.crmservice.entity.SysUser;
```

```java
import io.jsonwebtoken.*;
import org.springframework.stereotype.Component;

import java.util.Date;
import java.util.HashMap;
import java.util.Map;

@Component
public class JWTUtill {
    //JWT秘钥
    private String AUTHORIZE_TOKEN_SECRET = "cqut";
    //JWT过期时间，单位毫秒。 7*24*60*60*1000=604800000
    private long AUTHORIZE_TOKEN_EXPIRE = 604800000;

    public String createJwt(SysUser sysUser) {
        //jwt的加密算法
        SignatureAlgorithm signatureAlgorithm = SignatureAlgorithm.HS256;
        //获取当前时间戳,生成过期时间
        long nowMillis = System.currentTimeMillis();
        long expMillis = nowMillis + AUTHORIZE_TOKEN_EXPIRE;
        Date expDate = new Date(expMillis);
        //token的签发时间
        Date now = new Date(nowMillis);
        //需要保存到token字符串的有用信息
        Map<String, Object> map = new HashMap<>();
        map.put("suRole", sysUser.getSuRole());
        map.put("suId", "" + sysUser.getSuId());
        map.put("suName", sysUser.getSuName());
        JwtBuilder builder = Jwts.builder()
                .setClaims(map)  //设置附加信息
                // .setId("1")
//                .setSubject("权限验证")   // 主题
                .setIssuer("cn.edu.cqut")     // 签发者
                .setIssuedAt(now)      // 签发时间
                .signWith(signatureAlgorithm, AUTHORIZE_TOKEN_SECRET)  // 签名算法
以及密匙
                .setExpiration(expDate); // 过期时间
        return builder.compact();
    }

    /**
     * 验证JWT
     *
     * @param token
     * @return
     */

    public TableResult<Claims> validateJWT(String token) {
        Claims claims = null;
        try {
            claims = Jwts.parser()
                    .setSigningKey(AUTHORIZE_TOKEN_SECRET)
                    .parseClaimsJws(token)
                    .getBody();
            System.out.println("token是正确的");
            return TableResult.ok("", claims);
        } catch (ExpiredJwtException e) {
            System.out.println("token过期");
            return TableResult.error(2, "token过期");
        } catch (SignatureException e) {
            System.out.println("token签名不正确");
            return TableResult.error(3, "token校验异常");
        } catch (Exception e) {
            System.out.println("其他异常");
            return TableResult.error(4, "token异常");
        }
    }

    public static void main(String[] args) {
        // JWTUtill jwtUtill = new JWTUtill();
        // String jwt = jwtUtill.createJwt(null);
        //  System.out.println(jwt);
        //
        //
jwtUtill.validateJWT("eyJhbGciOiJIUzI1NiJ9.eyJyb2xlIjoiQURNSU4iLCJpc3MiOiJjb20ua
HF5aiIsImlkIjoiMTAwMSIsImV4cCI6MTY0ODg2NjYzMiwiaWF0IjoxNjQ4ODY2NjI3fQ.Erl0VDC9zJ
m-wENFbriSiTHP-jN3xBpodCFQTPskH3M");
        //
        //
    }
}
```

在SysUserController里面写登录方法

如果登录成功，生成token并返回给前端

```
import cn.edu.cqut.crmservice.entity.SysUser;
import cn.edu.cqut.crmservice.service.ISysUserService;
import cn.edu.cqut.crmservice.util.JWTUtill;
import cn.edu.cqut.crmservice.util.TableResult;
import com.baomidou.mybatisplus.core.conditions.query.QueryWrapper;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RestController;

/**
 * <p>
 *  前端控制器
 * </p>
 *
 * @author CQUT
 * @since 2023-06-10
 */
@RestController
@RequestMapping("/sysUser")
@CrossOrigin
public class SysUserController {
    @Autowired
    private ISysUserService sysUserService;
    @Autowired
    private JWTUtill jwtUtill;

    @PostMapping("/login")
    public TableResult<SysUser> login(SysUser sysUser){
        QueryWrapper<SysUser> wrapper = new QueryWrapper<>(sysUser);
        SysUser user = sysUserService.getOne(wrapper);
        if(user != null){
            String token = jwtUtill.createJwt(user);  //生成token
            return TableResult.ok(token, user);
        }else{
            return TableResult.error(1,"用户名或密码错误");
        }
    }
}
```

前端创建登录页面login.html

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title></title>
    <meta name="renderer" content="webkit">
    <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">
    <meta name="viewport" content="width=device-width, initial-scale=1, maximum-
scale=1">
    <meta name="apple-mobile-web-app-status-bar-style" content="black">
    <meta name="apple-mobile-web-app-capable" content="yes">
    <meta name="format-detection" content="telephone=no">
    <link rel="stylesheet" href="./layui/css/layui.css" media="all">
    <style>
        .loginBody form.layui-form {
            padding: 0 20px;
            width: 300px;
            height: 220px;
            position: absolute;
            left: 50%;
            top: 45%;
            margin: -150px 0 0 -150px;
            -webkit-box-sizing: border-box;
            -moz-box-sizing: border-box;
            -o-box-sizing: border-box;
            box-sizing: border-box;
            background: #fff;
            -webkit-border-radius: 5px;
            -moz-border-radius: 5px;
            border-radius: 5px;
            box-shadow: 0 0 50px #009688;
        }

        .loginbox-title {
            position: relative;
            text-align: center;
            width: 100%;
            height: 35px;
            padding-top: 10px;
            font-family: 'Lucida Sans', 'trebuchet MS', Arial, Helvetica;
            font-size: 20px;
            font-weight: normal;
            color: #444
        }
    </style>
</head>

<body class="loginBody">
    <form class="layui-form">
        <!-- <div class="login_face">
        <img src="../../images/Wjlogo.png" height="60px" class="userAvatar">
    </div> -->
        <div>
            <div class="loginbox-title">登录</div>
            <!-- <div class="loginbox-social">
            <div class="social-title">系统</div>
    </div> -->
            <div class="layui-form-item input-item">
                <input type="text" placeholder="请输入用户名" autocomplete="off"
id="username" class="layui-input">
            </div>
            <div class="layui-form-item input-item">
                <input type="password" placeholder="请输入密码" autocomplete="off"
id="password" class="layui-input">
            </div>
            <!-- <div class="layui-form-item input-item" id="imgCode">
        <input type="text" placeholder="请输入验证码" name="code"
autocomplete="off" id="code" class="layui-input">
        <img style="margin-right: 20px" width="100px" id="verify_code_img"
height="32"
             src="/verifyCode/getImg" onclick="javascript:changeImg()"><br>
    </div> -->
            <div class="layui-form-item  input-item">
                <button class="layui-btn" lay-filter="login" lay-submit
style="width:100%">登录</button>
            </div>

    </form>
</body>

<script type="text/javascript" src="./layui/layui.js"></script>
<script>
    layui.use(['jquery','form'], function () {
        var $ = layui.jquery;
        var form = layui.form
        form.on('submit(login)', function (data) {
            $.ajax({
                type: "post",
                url: "http://localhost:8080/sysUser/login",
                dataType: "json",
                data: {
                    suName: $("#username").val(),
                    suPwd: $("#password").val()
                },
                success: function (obj) {
                    if (obj.code == 0) {
                        localStorage.setItem("token", obj.msg);
                        localStorage.setItem("suName", obj.obj.suName)
                        location.href = "index.html"
                    } else {
                        layer.msg(obj.msg)
                    }

                }
            })
            return false;
        })


    });
</script>

</html>
```

修改首页

- 在head里面添加判断是否有token, 没有跳转到登录页面
- 右上角登录信息
- 在js代码把登录用户名显示在右上角

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1, maximum-
scale=1">
  <title>layout 管理系统大布局 - Layui</title>
  <link rel="stylesheet" href="./layui/css/layui.css">
  <script>
     if(localStorage.getItem("token")==undefined){
     window.location.href="login.html"
  }
  </script>
</head>
<body>
<div class="layui-layout layui-layout-admin">
  <div class="layui-header">
    <div class="layui-logo layui-hide-xs layui-bg-black">
        <img src="/img/logo.png" style="height: 40px;" alt="">
    </div>
    <!-- 头部区域（可配合layui 已有的水平导航） -->
    <ul class="layui-nav layui-layout-left">
      <li style="font-size: 24px;line-height: 60px;">CRM客户关系管理系统</li>
    </ul>
    <ul class="layui-nav layui-layout-right">
      <li class="layui-nav-item layui-hide layui-show-md-inline-block">
        <a href="javascript:;">
          <!-- <img
src="//tva1.sinaimg.cn/crop.0.0.118.118.180/5db11ff4gw1e77d3nqrv8j203b03cweg.jpg"
class="layui-nav-img"> -->
```

```
          <span id="userName"></span>
        </a>
        <dl class="layui-nav-child">
          <dd><a href="">用户信息</a></dd>
          <dd><a
href="javascript:localStorage.clear();location.href='login.html'">退出</a></dd>
        </dl>
      </li>
      <li class="layui-nav-item" lay-header-event="menuRight" lay-unselect>
        <a href="javascript:;">
          <i class="layui-icon layui-icon-more-vertical"></i>
        </a>
      </li>
    </ul>
  </div>

  <div class="layui-side layui-bg-black">
    <div class="layui-side-scroll">
      <!-- 左侧导航区域（可配合layui已有的垂直导航） -->
      <ul class="layui-nav layui-nav-tree" lay-filter="test">
        <li class="layui-nav-item layui-nav-itemed">
          <a class="" href="javascript:;">客户管理</a>
          <dl class="layui-nav-child">
            <!-- 页面在name叫content的iframe标签中显示 -->
            <dd><a href="customerlist.html" target="content">客户信息</a></dd>
            <dd><a href="contactlist.html" target="content">客户联系人</a></dd>
            <dd><a href="javascript:;">客户交往记录</a></dd>

          </dl>
        </li>
        <li class="layui-nav-item">
          <a href="javascript:;">营销管理</a>
          <dl class="layui-nav-child">
            <dd><a href="javascript:;">list 1</a></dd>
            <dd><a href="javascript:;">list 2</a></dd>
            <dd><a href="">超链接</a></dd>
          </dl>
        </li>
        <li class="layui-nav-item">
            <a href="javascript:;">服务管理</a>
            <dl class="layui-nav-child">
              <dd><a href="javascript:;">list 1</a></dd>
              <dd><a href="javascript:;">list 2</a></dd>
              <dd><a href="">超链接</a></dd>
            </dl>
          </li>
          <li class="layui-nav-item">
            <a href="javascript:;">统计报表</a>
            <dl class="layui-nav-child">
              <dd><a href="javascript:;">list 1</a></dd>
              <dd><a href="javascript:;">list 2</a></dd>
              <dd><a href="">超链接</a></dd>
            </dl>
          </li>
      </ul>
    </div>
  </div>
  <div class="layui-body">
    <!-- 内容主体区域 -->
    <div style="padding: 15px;">
        <iframe src="" name="content" style="width: 100%; height: 550px;"
frameborder="0"></iframe>
    </div>
  </div>

  <div class="layui-footer">
    <!-- 底部固定区域 -->
    重庆理工版权所有
  </div>
</div>
<script src="./layui/layui.js"></script>
<script>
//JS
layui.use(['element', 'layer', 'util'], function(){
  var element = layui.element
  ,layer = layui.layer
  ,util = layui.util
  ,$ = layui.$;

  //头部事件
  util.event('lay-header-event', {
    //左侧菜单事件
    menuLeft: function(othis){
      layer.msg('展开左侧菜单的操作', {icon: 0});
    }
    ,menuRight: function(){
      layer.open({
        type: 1
        ,content: '<div style="padding: 15px;">处理右侧面板的操作</div>'
        ,area: ['260px', '100%']
        ,offset: 'rt' //右上角
        ,anim: 5
        ,shadeClose: true
      });
    }
  });

  $("#userName").text(localStorage.getItem("suName"))

});
</script>
</body>
</html>
```

创建注解Auth

用于设置控制层的方法是否要限制访问权限

只要添加注解，values为true, 表示这个方法需要权限控制

roles参数设置允许访问的角色

示例： @Auth(value=true, roles="ADMIN, SALES")

```
import java.lang.annotation.ElementType;
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;
import java.lang.annotation.Target;

@Retention(RetentionPolicy.RUNTIME) //这个注解的作用域
@Target(ElementType.METHOD) //这个注解使用的位置
public @interface Auth {
    boolean value() default true;

    String roles() default "ADMIN";
}
```

创建拦截器

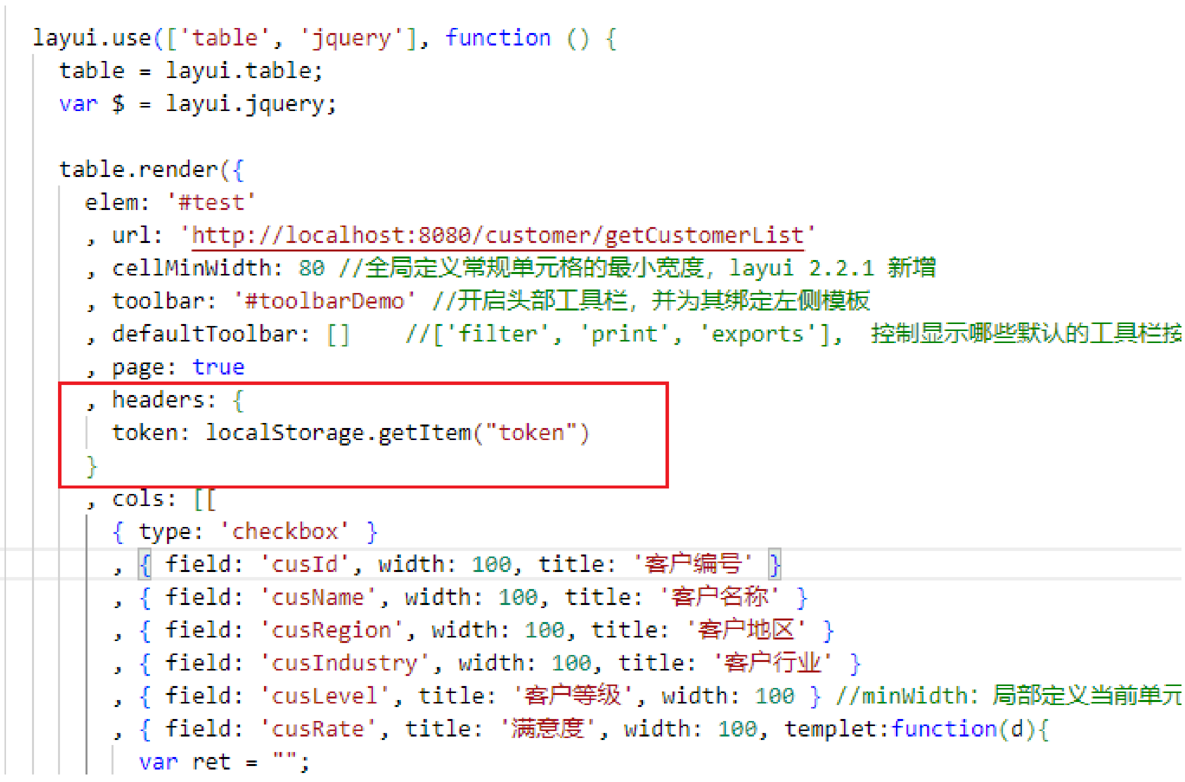
```
import java.lang.reflect.Method;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;


import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.method.HandlerMethod;
import org.springframework.web.servlet.HandlerInterceptor;

import com.alibaba.fastjson.JSON;


import io.jsonwebtoken.Claims;

public class AuthInterceptor implements HandlerInterceptor {
    @Autowired
    private JWTUtill jwtUtil;

    //获取token
    //校验token
    //根据token获取用户以及role
    //判断用户能否访问请求的这个方法
    @Override
    public boolean preHandle(HttpServletRequest httpServletRequest,
HttpServletResponse httpServletResponse, Object object)
            throws Exception {
        String token = httpServletRequest.getHeader("token");// 从 http 请求头中取
出 token
        // 如果不是映射到方法直接通过
        if (!(object instanceof HandlerMethod)) {
            return true;
        }
        HandlerMethod handlerMethod = (HandlerMethod) object;
        Method method = handlerMethod.getMethod();
        //检查有没有需要用户权限的注解
        if (method.isAnnotationPresent(Auth.class)) {
            Auth auth = method.getAnnotation(Auth.class);
            if (auth.value()) {
                // 没有提交token
                if (token == null) {
                    httpServletResponse.setCharacterEncoding("UTF-8");
                    httpServletResponse.setContentType("application/json;
charset=utf-8");
                    String respStr = JSON.toJSONString(TableResult.error(1,"没有
token，请重新登录"));

 httpServletResponse.getOutputStream().write(respStr.getBytes("UTF-8"));
                    return false;
                }
                //token校验失败
                TableResult<Claims> result = jwtUtil.validateJWT(token);  //校验
token
                if (result.getCode() != 0) {
                    httpServletResponse.setCharacterEncoding("UTF-8");
                    httpServletResponse.setContentType("application/json;
charset=utf-8");
                    String respStr = JSON.toJSONString(TableResult.error(2,
result.getMsg()));

 httpServletResponse.getOutputStream().write(respStr.getBytes("UTF-8"));
                    return false;
                }

                String suId = (String) result.getObj().get("suId");
                String suRole = (String) result.getObj().get("suRole");
                String suName = (String) result.getObj().get("suName");

                //没有权限
                if (!auth.roles().contains(suRole)) {
                    httpServletResponse.setCharacterEncoding("UTF-8");
                    httpServletResponse.setContentType("application/json;
charset=utf-8");
                    String respStr = JSON.toJSONString(TableResult.error(3,"没有
权限"));

 httpServletResponse.getOutputStream().write(respStr.getBytes("UTF-8"));
                    return false;
                }

                //把用户实体保存到request, 让控制层方法可以获取登录用户信息
                httpServletRequest.setAttribute("suRole", suRole);
                httpServletRequest.setAttribute("suName", suName);
                httpServletRequest.setAttribute("suId", suId);
            }
        }
        return true;
    }


}
```

拦截器的配置类InterceptorConfig

```java
import org.springframework.beans.factory.annotation.Configurable;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.InterceptorRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;


@Configuration
public class InterceptorConfig implements WebMvcConfigurer {
    @Override
    //设置要拦截的URL
    public void addInterceptors(InterceptorRegistry registry) {
        registry.addInterceptor(authenticationInterceptor())
                .addPathPatterns("/**")  //拦截所有请求
                .excludePathPatterns("/sysUser/login");  //不拦截的URL
    }

    @Bean   //把我们写的拦截器注入到容器
    public AuthInterceptor authenticationInterceptor() {
        return new AuthInterceptor();
    }
}
```

使用权限控制

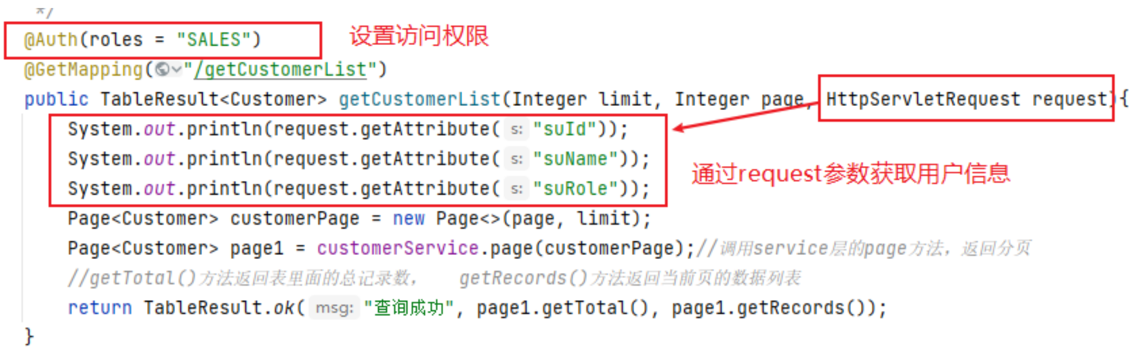
- 前端在发请求时在header里面提交token. 前端请求包括数据表格的render以及我们自己写的ajax()



```javascript
layui.use(['table', 'jquery'], function () {
  table = layui.table;
  var $ = layui.jquery;

  table.render({
    elem: '#test'
    , url: 'http://localhost:8080/customer/getCustomerList'
    , cellMinWidth: 80 //全局定义常规单元格的最小宽度, layui 2.2.1 新增
    , toolbar: '#toolbarDemo' //开启头部工具栏, 并为其绑定左侧模板
    , defaultToolbar: []    //['filter', 'print', 'exports'],  控制显示哪些默认的工具栏
    , page: true
    , headers: {
      token: localStorage.getItem("token")
    }
    , cols: [[
      { type: 'checkbox' }
      , { field: 'cusId', width: 100, title: '客户编号' }
      , { field: 'cusName', width: 100, title: '客户名称' }
      , { field: 'cusRegion', width: 100, title: '客户地区' }
      , { field: 'cusIndustry', width: 100, title: '客户行业' }
      , { field: 'cusLevel', title: '客户等级', width: 100 } //minWidth: 局部定义当前单元
      , { field: 'cusRate', title: '满意度', width: 100, templet:function(d){
        var ret = "";
```

- 控制层的方法添加@Auth注解，roles参数就是允许访问的角色

控制层的方法加一个HttpServletRequest类型的参数，通过这个参数的getAttribute()方法可以获取登录用户的id, 账号和角色，这些信息是在拦截器里面保存的



```java
*/
@Auth(roles = "SALES")
@GetMapping(⊙∨"/getCustomerList")
public TableResult<Customer> getCustomerList(Integer limit, Integer page, HttpServletRequest request){
    System.out.println(request.getAttribute( s: "suId"));
    System.out.println(request.getAttribute( s: "suName"));
    System.out.println(request.getAttribute( s: "suRole"));
    Page<Customer> customerPage = new Page<>(page, limit);
    Page<Customer> page1 = customerService.page(customerPage);//调用service层的page方法，返回分页
    //getTotal()方法返回表里面的总记录数，    getRecords()方法返回当前页的数据列表
    return TableResult.ok( msg: "查询成功", page1.getTotal(), page1.getRecords());
}
```

用ECharts实现统计图表

ECharts是一个开源的JavaScript统计图表库，可以在网页上实现柱状图、曲线去、饼图等各种统计图

ECharts官网: https://echarts.apache.org/

ECharts快速上手

- 下载echarts.js库文件，放到前端项目的js目录下
- 在网页中导入echarts.js
- 网页中添加一个div
- 创建echarts对象，设置参数

完整页面如下：

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

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <link rel="stylesheet" href="./layui/css/layui.css">
</head>

<body>
    <div id="main" style="width: 600px;height:400px;"></div>

    <script src="./layui/layui.js"></script>
    <script src="./js/echarts.js"></script>
    <script>
        layui.use(['jquery'], function () {
            var $ = layui.jquery;

            // 基于准备好的dom，初始化echarts实例
            var myChart = echarts.init(document.getElementById('main'));

            // 指定图表的配置项和数据
            var option = {
                title: {
                    text: '按地区统计客户数量' //统计图的标题
                },
                tooltip: {},   //鼠标放到柱子上提示，没有设置表示用默认提示
                legend: {
                    data: ['数量']   //图例，每一种颜色柱子代表的意义
                },
                xAxis: {
                    data: ['衬衫', '羊毛衫', '雪纺衫', '裤子', '高跟鞋', '袜子']   //
横坐标上的商品类别
                },
                yAxis: {},   //纵坐标上的刻度，没有设置就按默认方式标注刻度
                series: [
                    {
                        name: '数量',   //与图例对应
                        type: 'bar',   //统计图类型：  bar-柱状图； line-折线图; pie-
饼图
                        data: [5, 20, 36, 10, 10, 20]    //每根柱子的数值
                    }
                ]
            };
            // 使用刚指定的配置项和数据显示图表。
            myChart.setOption(option);
        });
    </script>
</body>

</html>
```

按地区统计客户数量报表实现

后台

- 创建用于接收查询返回数据的实体Report

```
package cn.edu.cqut.crmservice.entity;

import java.io.Serializable;

public class Report implements Serializable {
    private String item;
    private Long value;

    public String getItem() {
        return item;
    }

    public void setItem(String item) {
        this.item = item;
    }

    public Long getValue() {
        return value;
    }

    public void setValue(Long value) {
        this.value = value;
    }
}
```

- 创建给前端返回JSON的实体ReportResult

```
package cn.edu.cqut.crmservice.util;

import java.util.List;

public class ReportResult{
    private List<String> items;
    private List<Long> values;

    public ReportResult(List<String> items, List<Long> values) {
        this.items = items;
        this.values = values;
    }

    public static ReportResult ok(List<String> items, List<Long> values){
        return new ReportResult(items, values);
    }

    public List<String> getItems() {
        return items;
    }

    public void setItems(List<String> items) {
        this.items = items;
    }

    public List<Long> getValues() {
        return values;
    }

    public void setValues(List<Long> values) {
        this.values = values;
    }
}
```

- CustomerMapper中添加查询接口方法和要执行的SQL语句

```

    @Select("select count(*) value, cus_region item from customer GROUP BY
cus_region")
    public List<Report> getCustomerCountByRegion();
```

- ICustomerService中添加对应的方法声明

```
public List<Report> getCustomerCountByRegion();
```

- CustomerServiceImpl中实现该方法

baseMapper是父类中的mapper对象

```
    public List<Report> getCustomerCountByRegion() {
        return baseMapper.getCustomerCountByRegion();
    }
```

- 创建一个ReportController类

在方法中调用自定义的查询方法查询统计数据，然后按前端的要求转换成ReportResult需要的格式

```
package cn.edu.cqut.crmservice.controller;


import cn.edu.cqut.crmservice.entity.Report;
import cn.edu.cqut.crmservice.service.ICustomerService;
import cn.edu.cqut.crmservice.util.ReportResult;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import java.util.ArrayList;
import java.util.List;

@RestController  //给前端返回json数据
@RequestMapping("/report")
@CrossOrigin //允许跨域请求
public class ReportController {
    @Autowired
    private ICustomerService customerService;

    @GetMapping("/getCustomerCountByRegion")
    public ReportResult getCustomerCountByRegion(){
        List<Report> reports = customerService.getCustomerCountByRegion();
        List<String> items = new ArrayList<>();
        List<Long> values = new ArrayList<>();
        for (Report report : reports) {
            items.add(report.getItem());
            values.add(report.getValue());
        }
        return ReportResult.ok(items, values);
    }

}
```

前端

- 改造报表页面，通过ajax调用后台的报表接口方法，根据返回数据重新设置echarts相关参数

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

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <link rel="stylesheet" href="./layui/css/layui.css">
</head>

<body>
    <div id="main" style="width: 600px;height:400px;"></div>

    <script src="./layui/layui.js"></script>
    <script src="./js/echarts.js"></script>
    <script>
        layui.use(['jquery'], function () {
            var $ = layui.jquery;

            // 基于准备好的dom，初始化echarts实例
            var myChart = echarts.init(document.getElementById('main'));

            // 指定图表的配置项和数据
            var option = {
                title: {
                    text: '按地区统计客户数量' //统计图的标题
                },
                tooltip: {},   //鼠标放到柱子上提示，没有设置表示用默认提示
                legend: {
                    data: ['数量']   //图例，每一种颜色柱子代表的意义
                },
                xAxis: {
                    data: []   //横坐标上的商品类别
                },
                yAxis: {},   //纵坐标上的刻度，没有设置就按默认方式标注刻度
                series: [
                    {
                        name: '数量',   //与图例对应
                        type: 'bar',   //统计图类型：  bar-柱状图； line-折线图; pie-
饼图

                        data: []    //每根柱子的数值
                    }
                ]
            };

            // 使用刚指定的配置项和数据显示图表。
            myChart.setOption(option);

            $.ajax({
                type: "get",
                url: "http://localhost:8080/report/getCustomerCountByRegion",
                dataType: "json",
                success: function (obj) { //后台方法成功执行并返回结果时，会调用这个方
法，参数是后台返回的内容
                    myChart.setOption({
                        xAxis: {
                            data: obj.items   //横坐标上的商品类别
                        },
                        series: [
                            {
                                name: '数量',   //与图例对应
                                type: 'bar',   //统计图类型：  bar-柱状图； line-折线
图; pie-饼图
```

```
                                    data: obj.values    //每根柱子的数值
                                }
                            ]
                        });
                    }
                })

            });
        </script>
    </body>

    </html>
```

自定义查询方法支持分页，QueryWrapper，关联查询

- Contact实体中添加关联属性Customer, 以及getter, setter

```
private Customer customer;

  public Customer getCustomer() {
        return customer;
    }

    public void setCustomer(Customer customer) {
        this.customer = customer;
    }
```

- mapper接口

```
    @Select("select * from contact ${ew.customSqlSegment}")
    @Results({
//            cus_id用于关联查询后，原来的cusId属性不会有值，为了让它有值需要重新映射一遍
            @Result(column = "cus_id", property = "cusId"),
//            多对一关联查询
            @Result(column = "cus_id", property = "customer",
//                    多对一用One， 一对多用many.  select参数是根据关联字段查询关联对象的
mapper方法
                    one = @One(select =
"cn.edu.cqut.crmservice.mapper.CustomerMapper.selectById", fetchType =
FetchType.EAGER))
    })
    public Page<Contact> myPage(IPage<Contact> page, @Param(Constants.WRAPPER)
QueryWrapper<Contact> queryWrapper);
```

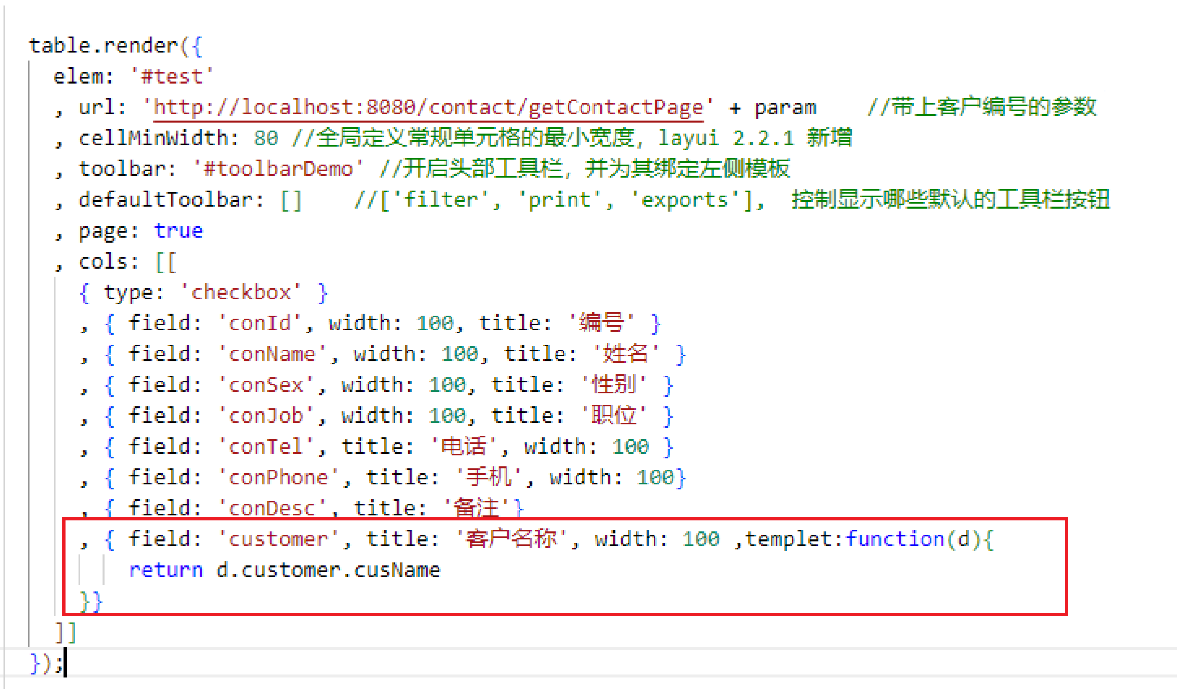
- service接口也定义相应方法

```
public Page<Contact> myPage(IPage<Contact> page, QueryWrapper<Contact>
queryWrapper);
```

- service的实现类中实现方法

```
    public Page<Contact> myPage(IPage<Contact> page, QueryWrapper<Contact>
queryWrapper) {
        return baseMapper.myPage(page, queryWrapper);
    }
```

- 前端页面用templet显示关联属性的值



```
table.render({
  elem: '#test'
  , url: 'http://localhost:8080/contact/getContactPage' + param    //带上客户编号的参数
  , cellMinWidth: 80 //全局定义常规单元格的最小宽度, layui 2.2.1 新增
  , toolbar: '#toolbarDemo' //开启头部工具栏, 并为其绑定左侧模板
  , defaultToolbar: []    //['filter', 'print', 'exports'], 控制显示哪些默认的工具栏按钮
  , page: true
  , cols: [[
    { type: 'checkbox' }
    , { field: 'conId', width: 100, title: '编号' }
    , { field: 'conName', width: 100, title: '姓名' }
    , { field: 'conSex', width: 100, title: '性别' }
    , { field: 'conJob', width: 100, title: '职位' }
    , { field: 'conTel', title: '电话', width: 100 }
    , { field: 'conPhone', title: '手机', width: 100}
    , { field: 'conDesc', title: '备注'}
    , { field: 'customer', title: '客户名称', width: 100 ,templet:function(d){
        return d.customer.cusName
    }}
  ]]
});
```