第一章 项目底层架构

1. 项目介绍

乐购商城是类似于京东商城，是一个全品类电商项目，后台实现，商品，分类，规格，品牌，以及用户角色权限的管理，前端查询商品，加入购物车，创建订单，第三方支付，以及商品秒杀功能。从0到1，掌握微服务架构、分布式、vue、全栈开发，以及电商千万级并发场景，解决方案。

1.2 技术栈

- Spring Boot 2.3.2.RELEASE
- Spring Cloud Hoxton.SR8
- Spring Cloud Alibaba 2.2.5.RELEASE
- Mybatis 3.5.4
- Mybatis Plus 3.3.2
- Spring Security OAuth2 2.2.4.RELEASE
- Vue 2.5.10
- Vuex 3.0.1
- IView 3.2.2

1.3 架构图

架构图水平面上的业务模块，加上垂直面上的技术模块，互相依赖形成的逻辑结构图，如图1–1所示。

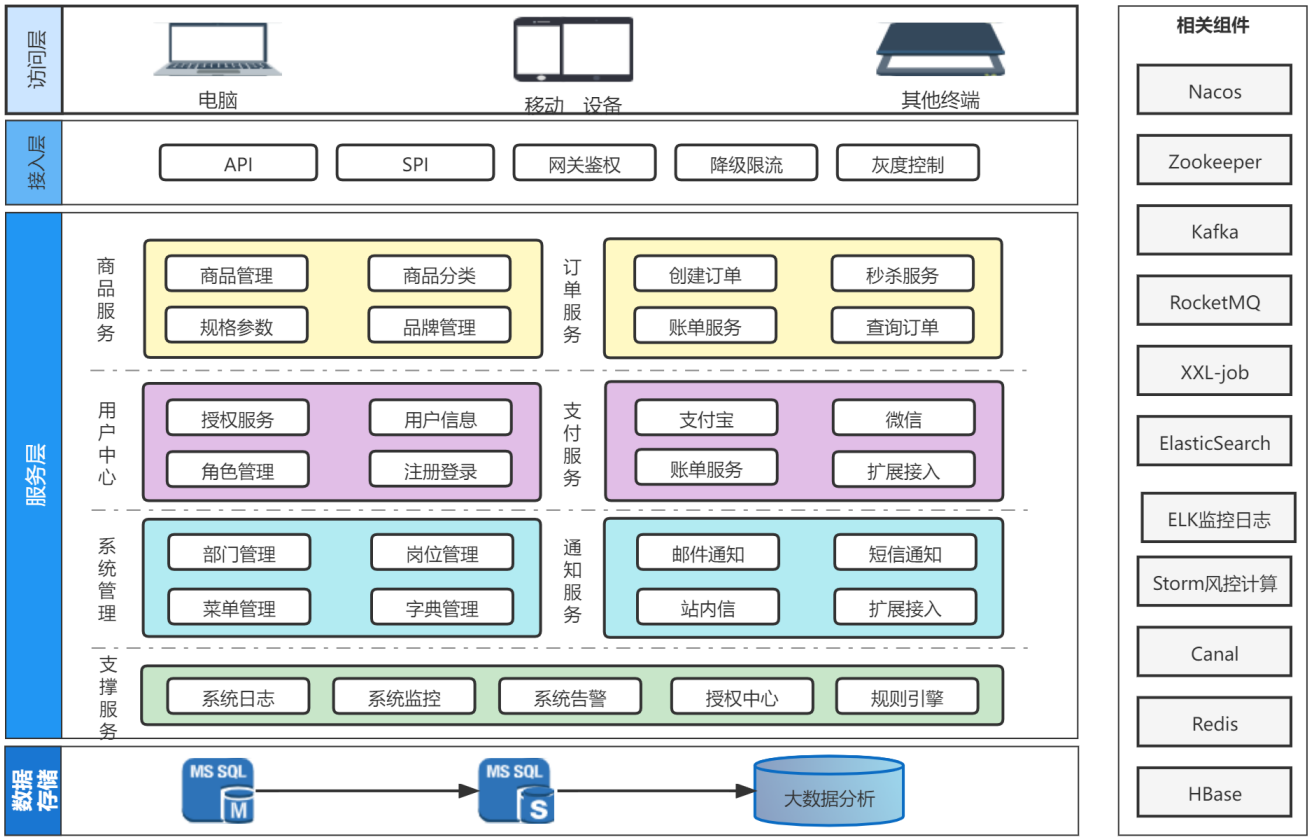

图1–1 乐购商城架构图

2. 搭建项目

2.1 项目基本结构

项目分为前端项目和后端项目，基本工程分类如下图2–1所示

图2–1 项目工程结构分类

2.2 项目启动顺序

1. 后端项目启动顺序
   先启动授权中心（负责颁发和校验令牌），其他微服务后启动
2. 前端项目启动

XML 复制代码

```xml
npm install (yarn)
npm run dev (yarn dev)
```

2.2 搭建前端项目

项目前端组件使用Vue和IView开发，这里时间关系就不带领大家使用vue cli依次从头创建前端项目，大家可以直接使用，素材提供的前端项目，开发后续业务应用，**注意前端项目虽然提供提供底层架构代码，业务代码，但是还是推荐大家根据业务自行开发**，同时也会讲解典型前端业务代码实现。前端工程结构如图1–2所示。

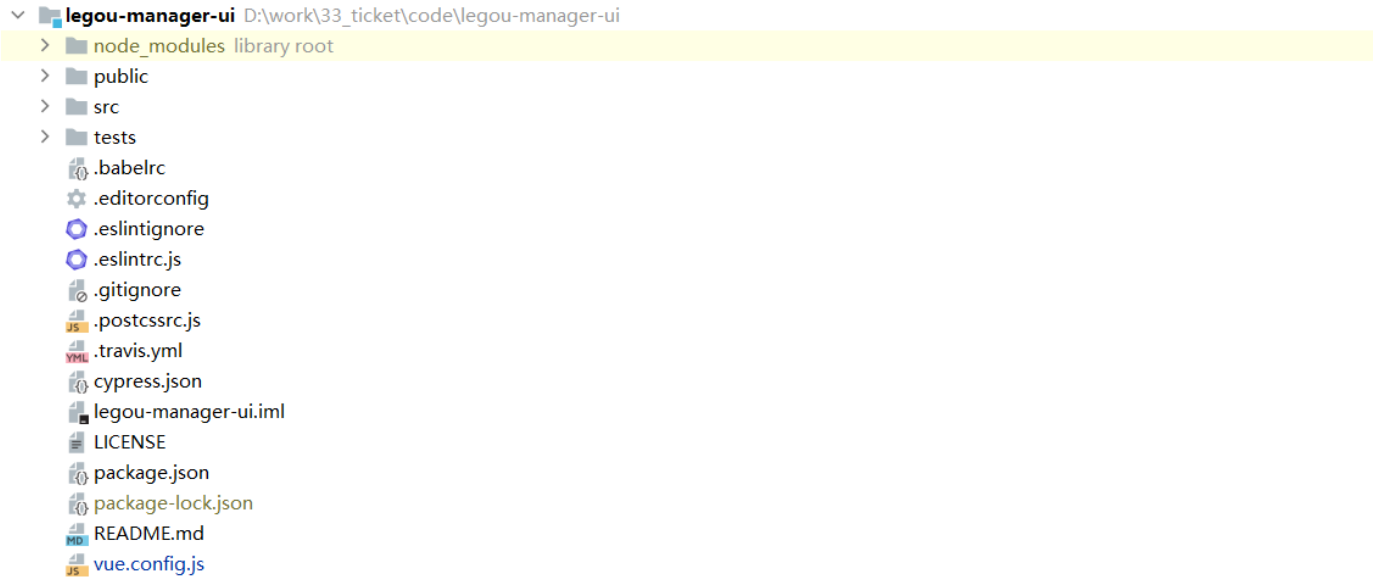

图1–2 前端项目结构

从素材中拷贝前端项目后，运行`cnpm install`命令，安装依赖组件

运行`npm run dev`，或者在idea中配置如图1–3所示，运行前端项目

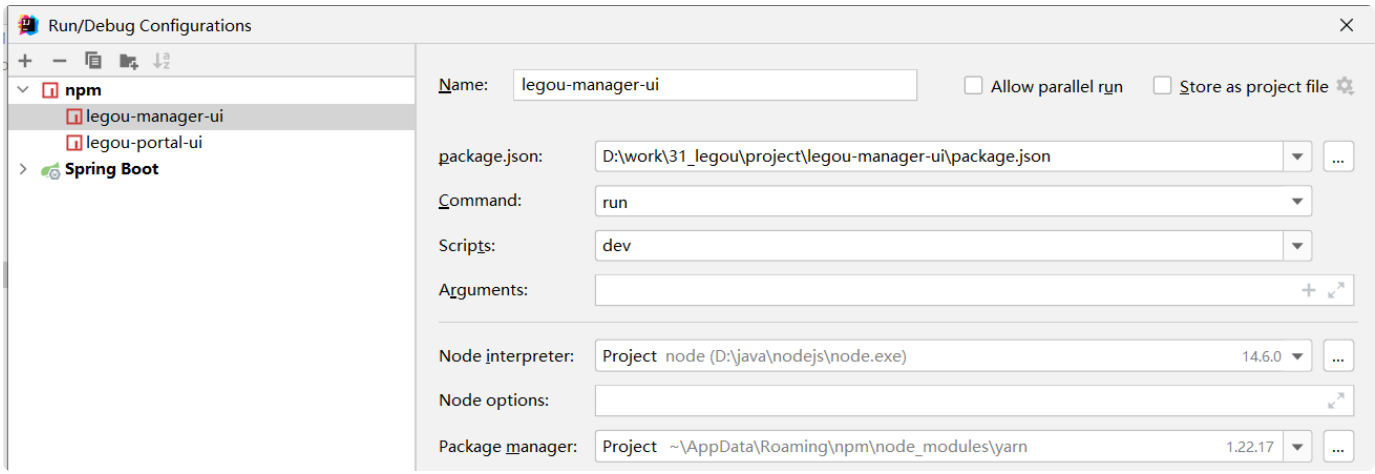

图1–3 idea运行vue项目

前端去掉认证授权代码修改如下。
src\components\main\main.vue

Java 复制代码

```
    menuList () {
      //return this.$store.getters.menuList
    },
```

src\router\index.js

```javascript
router.beforeEach((to, from, next) => {
  iView.LoadingBar.start()
  const token = getToken()
  next()
})
```

2.4 搭建后端项目

2.4.1 Nacos

项目采用Nacos作为注册中心和配置中心，Nacos是阿里巴巴开源的一款支持服务注册与发现，配置管理以及微服务管理的组件。用来取代以前常用的注册中心（ZooKeeper，Eureka等），以及配置中心（Spring Cloud Config等）。Nacos是集成了注册中心和配置中心的功能。

在使用Nacos之前需要先下载Nacos Server，下载地址：https://github.com/alibaba/nacos/releases/download/1.4.1/nacos-server-1.4.1.zip。

启动Nacos命令如下。

```java
bin/startup.sh -m standalone # linux
bin/startup.cmd -m standalone # windows
```

或者修改配置文件startup.cmd，代码如下，默认为cluster，然后直接运行startup.cmd

```xml
set MODE="standalone"
```

然后访问http://localhost:8848/nacos，进入nacos管控台，默认账号密码为nacos/nacos，如图1-4所示。

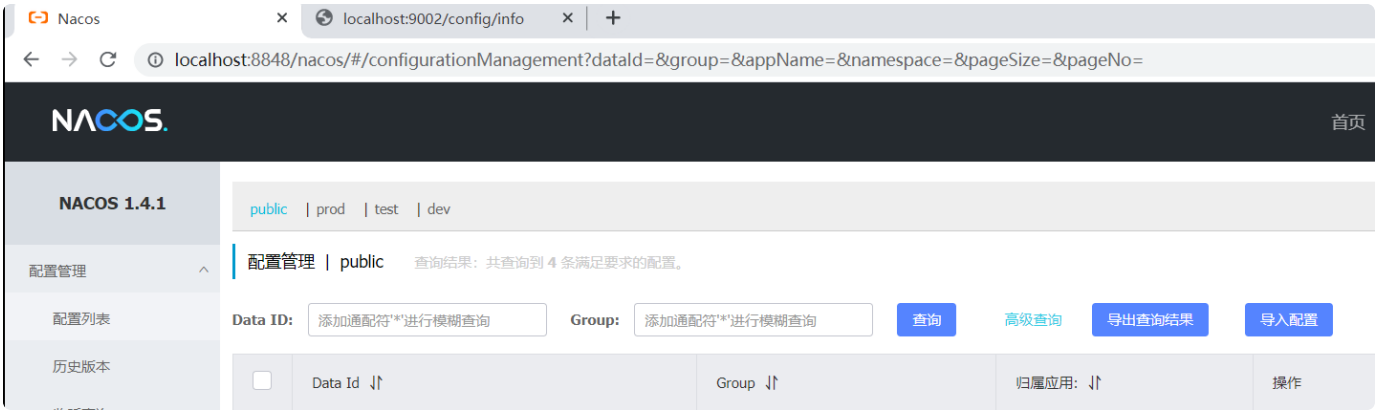

图1-4 Nacos管控台

导入素材提供的配置到Nacos，具体业务用的的配置，在业务中在详细讲解，Nacos配置如图1-5所示。

Data Id	Group	归属应用:	操作
common.yaml	DEFAULT_GROUP	公共配置（注册中心等）	详情 | 示例代码 | 编辑 | 删除 | 更多
db.yaml	DEFAULT_GROUP	数据库参数配置	详情 | 示例代码 | 编辑 | 删除 | 更多
security.yaml	DEFAULT_GROUP	需要授权的资源中心配置	详情 | 示例代码 | 编辑 | 删除 | 更多
auth-center-dev.yaml	DEFAULT_GROUP	授权中心	详情 | 示例代码 | 编辑 | 删除 | 更多
gateway-dev.yaml	DEFAULT_GROUP	网关	详情 | 示例代码 | 编辑 | 删除 | 更多
admin-service-dev.yaml	DEFAULT_GROUP	系统设置	详情 | 示例代码 | 编辑 | 删除 | 更多
canal-service-dev.yaml	DEFAULT_GROUP	canal微服务配置	详情 | 示例代码 | 编辑 | 删除 | 更多
item-service-dev.yaml	DEFAULT_GROUP	商品微服务配置	详情 | 示例代码 | 编辑 | 删除 | 更多
order-service-dev.yaml	DEFAULT_GROUP	订单微服务配置	详情 | 示例代码 | 编辑 | 删除 | 更多
page-service-dev.yaml	DEFAULT_GROUP	商品详情静态化配置	详情 | 示例代码 | 编辑 | 删除 | 更多
pay-service-dev.yaml	DEFAULT_GROUP	支付服务配置	详情 | 示例代码 | 编辑 | 删除 | 更多
search-service-dev.yaml	DEFAULT_GROUP	搜索微服务配置	详情 | 示例代码 | 编辑 | 删除 | 更多
security-service-dev.yaml	DEFAULT_GROUP	用户中心配置	详情 | 示例代码 | 编辑 | 删除 | 更多

图1-5 Nacos中的配置

2.4.2 父工程

创建legou-parent统一管理聚合子工程的依赖构件的版本号，pom.xml如下所示。

- spring boot 2.3.2.RELEASE
- spring cloud Hoxton.SR8

- spring cloud albiaba 2.2.5.RELEASE

XML 复制代码

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.lxs</groupId>
    <artifactId>legou-parent</artifactId>
    <version>1.0-SNAPSHOT</version>
    <modules>
        <module>legou-auth-center</module>
        <module>legou-core</module>
        <module>legou-admin</module>
        <module>legou-gateway</module>
        <module>legou-security</module>
        <module>legou-upload</module>
        <module>legou-item</module>
        <module>legou-search</module>
        <module>legou-common</module>
        <module>legou-canal</module>
        <module>legou-page</module>
        <module>legou-order</module>
        <module>legou-pay</module>
        <module>legou-seckill</module>
    </modules>

    <packaging>pom</packaging>

    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.3.2.RELEASE</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>

    <properties>
        <java.version>1.8</java.version>
        <alibaba-cloud.version>2.2.5.RELEASE</alibaba-cloud.version>
        <springcloud.version>Hoxton.SR8</springcloud.version>
    </properties>

    <dependencyManagement>
        <dependencies>
            <dependency>
                <groupId>org.springframework.cloud</groupId>
                <artifactId>spring-cloud-dependencies</artifactId>
                <version>${springcloud.version}</version>
```

```
                <type>pom</type>
                <scope>import</scope>
            </dependency>

            <dependency>
                <groupId>com.alibaba.cloud</groupId>
                <artifactId>spring-cloud-alibaba-dependencies</artifactId>
                <version>${alibaba-cloud.version}</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
        </dependencies>
    </dependencyManagement>

    <dependencies>
        <dependency>
            <groupId>org.apache.commons</groupId>
            <artifactId>commons-lang3</artifactId>
            <version>3.9</version>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>


</project>
```

2.4.3 基类工程

创建legou-core基类工程，主要实现dao，service，controller基类，和项目用到的工具类，比如雪花算法工具类等。直接拷贝使用，具体代码复用实现下一个小节详细讲解。

2.4.4 微服务网关

创建legou-gateway网关工程。

1. pom.xml

XML 复制代码

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

    <parent>
        <artifactId>legou-parent</artifactId>
        <groupId>com.lxs</groupId>
        <version>1.0-SNAPSHOT</version>
    </parent>
    <modelVersion>4.0.0</modelVersion>

    <artifactId>legou-gateway</artifactId>

    <dependencies>
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-gateway</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-openfeign</artifactId>
        </dependency>
        <!--SpringCloud ailibaba nacos -->
        <dependency>
            <groupId>com.alibaba.cloud</groupId>
            <artifactId>spring-cloud-starter-alibaba-nacos-discovery</artifactId>
        </dependency>
        <!--nacos-config-->
        <dependency>
            <groupId>com.alibaba.cloud</groupId>
            <artifactId>spring-cloud-starter-alibaba-nacos-config</artifactId>
        </dependency>
        <!--SpringCloud ailibaba sentinel -->
        <dependency>
            <groupId>com.alibaba.cloud</groupId>
            <artifactId>spring-cloud-starter-alibaba-sentinel</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>

```

```
        <!--redis-->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-redis-reactive</artifactId>
            <version>2.1.3.RELEASE</version>
        </dependency>

        <!--鉴权-->
        <dependency>
            <groupId>io.jsonwebtoken</groupId>
            <artifactId>jjwt</artifactId>
            <version>0.9.0</version>
        </dependency>

        <dependency>
            <groupId>com.lxs</groupId>
            <artifactId>legou-security-instance</artifactId>
            <version>${project.version}</version>
            <exclusions>
                <exclusion>
                    <groupId>org.springframework.boot</groupId>
                    <artifactId>spring-boot-starter-jdbc</artifactId>
                </exclusion>
                <exclusion>
                    <groupId>org.springframework.boot</groupId>
                    <artifactId>spring-boot-starter-web</artifactId>
                </exclusion>
            </exclusions>
        </dependency>


    </dependencies>

</project>
```

2. 配置文件

application.yml

YAML 复制代码

```yaml
spring:
  application:
    name: gateway
  profiles:
    active: dev
```

bootstrap.yml

YAML 复制代码

```yaml
spring:
  cloud:
    nacos:
      config:
        server-addr: localhost:8848
        file-extension: yaml
        extension-configs[0]:
          data-id: common.yaml
          refresh: true
  # 多个接口上的@FeignClient(“相同服务名”)会报错，overriding is disabled。
  # 设置 为true ,即 允许 同名
  main:
    allow-bean-definition-overriding: true
```

3. 启动器

Java 复制代码

```java
package com.lxs.cloud;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.client.circuitbreaker.EnableCircuitBreaker;
import org.springframework.cloud.client.discovery.EnableDiscoveryClient;
import org.springframework.cloud.gateway.filter.ratelimit.KeyResolver;
import org.springframework.cloud.openfeign.EnableFeignClients;
import org.springframework.context.annotation.Bean;
import org.springframework.web.server.ServerWebExchange;
import reactor.core.publisher.Mono;

/**
 * @author zhengweimin
 */
@SpringBootApplication
@EnableDiscoveryClient
@EnableFeignClients
@EnableCircuitBreaker
public class GatewayApplication {

    public static void main(String[] args) {
        SpringApplication.run(GatewayApplication.class, args);
    }

}
```

2.4.4 系统管理微服务

创建legou-admin，聚合父工程，在legou-admin下创建legou-admin-interface，存放接口和实体类，创建legou-admin-service，存放具体dao，service，controller实现类，这样如果其他工程只需要依赖票务管理的接口实体类，只需要引入legou-admin-interface即可。

1. legou-admin

XML 复制代码

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <parent>
        <artifactId>legou-parent</artifactId>
        <groupId>com.lxs</groupId>
        <version>1.0-SNAPSHOT</version>
    </parent>
    <modelVersion>4.0.0</modelVersion>
    <artifactId>legou-admin</artifactId>
    <packaging>pom</packaging>
    <modules>
        <module>legou-admin-instance</module>
        <module>legou-admin-service</module>
    </modules>

</project>
```

2. legou-admin-interface

XML 复制代码

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <parent>
        <artifactId>legou-admin</artifactId>
        <groupId>com.lxs</groupId>
        <version>1.0-SNAPSHOT</version>
    </parent>
    <modelVersion>4.0.0</modelVersion>

    <artifactId>legou-admin-instance</artifactId>

    <dependencies>

        <dependency>
            <groupId>com.lxs</groupId>
            <artifactId>legou-core</artifactId>
            <version>${project.version}</version>
        </dependency>

        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <scope>provided</scope>
        </dependency>


    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
                <configuration>
                    <skip>true</skip>
                </configuration>
            </plugin>
        </plugins>
    </build>


</project>
```

这里要强调<skip>true<skip>，因为此工程也是从父工程继承，也就依赖了spring boot，但是此工程只是提供接口实体类，给其他工程使用，不提供spring boot的启动器，所以必须配置<skip>true</skip>否则，maven构建时，因为找不到main方法报错。

3. legou-admin-service

（1）pom.xml

XML 复制代码

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <parent>
        <artifactId>legou-admin</artifactId>
        <groupId>com.lxs</groupId>
        <version>1.0-SNAPSHOT</version>
    </parent>
    <modelVersion>4.0.0</modelVersion>

    <artifactId>legou-admin-service</artifactId>

    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-actuator</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-openfeign</artifactId>
        </dependency>

        <!--SpringCloud ailibaba nacos -->
        <dependency>
            <groupId>com.alibaba.cloud</groupId>
            <artifactId>spring-cloud-starter-alibaba-nacos-discovery</artifactId>
        </dependency>
        <!--nacos-config-->
        <dependency>
            <groupId>com.alibaba.cloud</groupId>
            <artifactId>spring-cloud-starter-alibaba-nacos-config</artifactId>
        </dependency>
        <!--SpringCloud ailibaba sentinel -->
        <dependency>
            <groupId>com.alibaba.cloud</groupId>
```

```
46              <artifactId>spring-cloud-starter-alibaba-sentinel</artifactId>
47          </dependency>
48
49          <dependency>
50              <groupId>com.lxs</groupId>
51              <artifactId>legou-core</artifactId>
52              <version>${project.version}</version>
53          </dependency>
54
55          <dependency>
56              <groupId>org.projectlombok</groupId>
57              <artifactId>lombok</artifactId>
58              <scope>provided</scope>
59          </dependency>
60
61          <dependency>
62              <groupId>mysql</groupId>
63              <artifactId>mysql-connector-java</artifactId>
64              <version>5.1.46</version>
65              <scope>runtime</scope>
66          </dependency>
67
68          <dependency>
69              <groupId>com.lxs</groupId>
70              <artifactId>legou-admin-instance</artifactId>
71              <version>${project.version}</version>
72          </dependency>
73
74          <!-- swagger -->
75          <dependency>
76              <groupId>io.springfox</groupId>
77              <artifactId>springfox-swagger2</artifactId>
78              <version>2.9.2</version>
79          </dependency>
80          <dependency>
81              <groupId>io.springfox</groupId>
82              <artifactId>springfox-swagger-ui</artifactId>
83              <version>2.9.2</version>
84          </dependency>
85
86          <!--oauth2-->
87          <dependency>
88              <groupId>org.springframework.cloud</groupId>
89              <artifactId>spring-cloud-starter-oauth2</artifactId>
90          </dependency>
91
92      </dependencies>
93
94  </project>
```

（2）启动器

Java 复制代码

```java
package com.lxs.legou;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.client.circuitbreaker.EnableCircuitBreaker;
import org.springframework.cloud.client.discovery.EnableDiscoveryClient;
import org.springframework.cloud.openfeign.EnableFeignClients;

@SpringBootApplication
@EnableDiscoveryClient
@EnableFeignClients
@EnableCircuitBreaker
public class AdminApplication {

    public static void main(String[] args) {
        SpringApplication.run(AdminApplication.class, args);
    }

}
```

（3）配置文件

application.yml

YAML 复制代码

```yaml
spring:
  application:
    name: admin-service
  profiles:
    active: dev
```

bootstrap.yml

YAML 复制代码

```yaml
spring:
  cloud:
    nacos:
      config:
        server-addr: localhost:8848
        file-extension: yaml
        extension-configs[0]:
          data-id: common.yaml
          refresh: true
        extension-configs[1]:
          data-id: db.yaml
          refresh: true
        extension-configs[2]:
          data-id: security.yaml
          refresh: true
  # 多个接口上的@FeignClient(“相同服务名”)会报错，overriding is disabled。
  # 设置 为true ,即 允许 同名
  main:
    allow-bean-definition-overriding: true
```

3.1 代码复用

3.1 后端复用

借鉴MybatisPlus的思想，通过po、dao、service、controller继承各自的基类，这样具体业务的CRUD代码就不需要在写代码了，各自继承相应的基类即可，基类代码如下。

3.1.1 实体类基类

1. BaseEntity

   BaseEntity是所有实体类的基类，类图如图1-6所示。

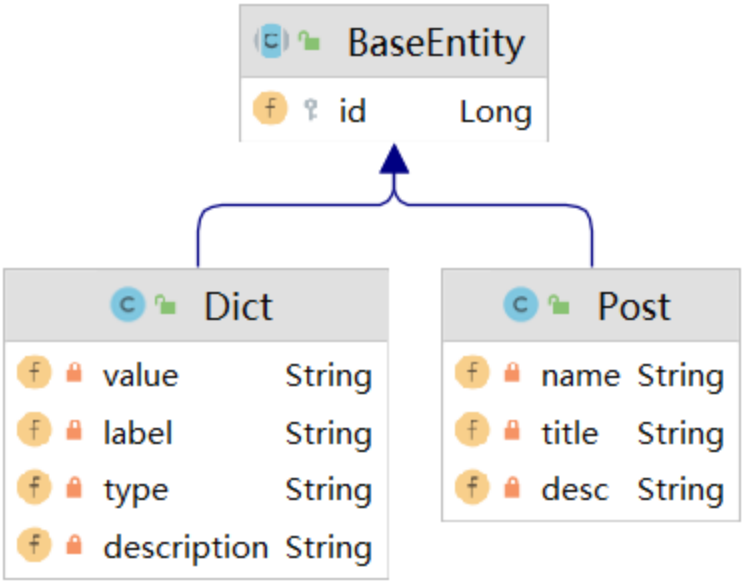

图1-6 BaseEntity类图

所有实体类都继承此基类，比如用户，岗位，角色等，代码如下。

Java 复制代码

```java
@Data
@JsonIgnoreProperties(value = {"handler"})
public abstract class BaseEntity implements Serializable {

  /**
   * 实体编号（唯一标识）
   */
  @TableId(value = "`id`", type = IdType.AUTO)
  protected Long id;

}
```

2. BaseTreeEntity

BaseTreeEntity是所有树结构数据的基类，类图如图1-7所示。

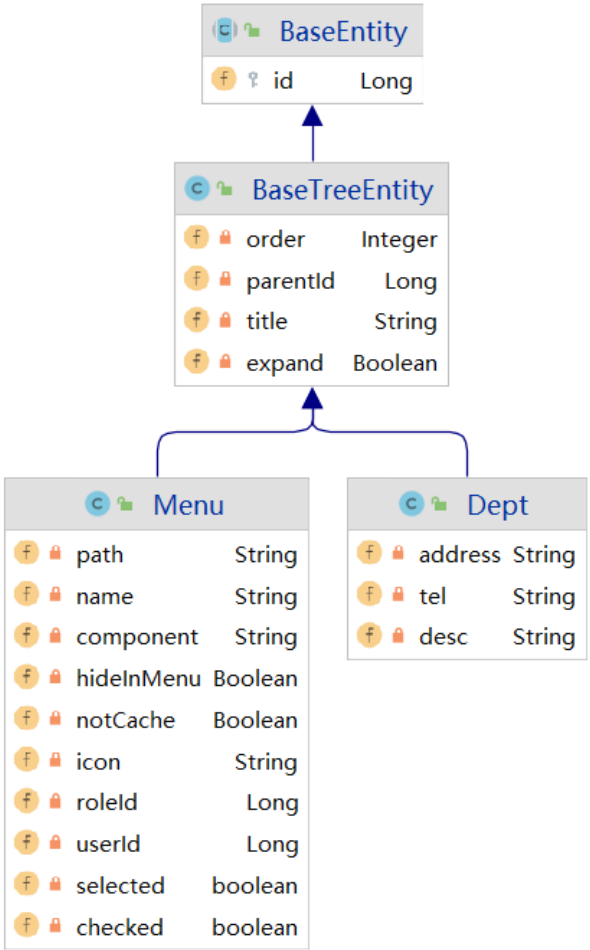

图1–7 BaseTreeEntity

树结构数据都继承BaseTreeEntity，比如部门，菜单等，代码如下所示。

Java 复制代码

```java
@Data
@JsonIgnoreProperties(value = {"handler"})
public class BaseTreeEntity extends BaseEntity {

    /**
     * 排序字段
     */
    @TableField("`order`")
    private Integer order;

    /**
     * 父节点id
     */
    @TableField("`parent_id`")
    private Long parentId;

    /**
     * 节点名称
     */
    @TableField("`title`")
    private String title;

    /**
     * 是否展开节点
     */
    @TableField("`expand`")
    private Boolean expand = false;

}
```

3.1.2 Dao的基类

Dao基类，类图如图1-8所示。

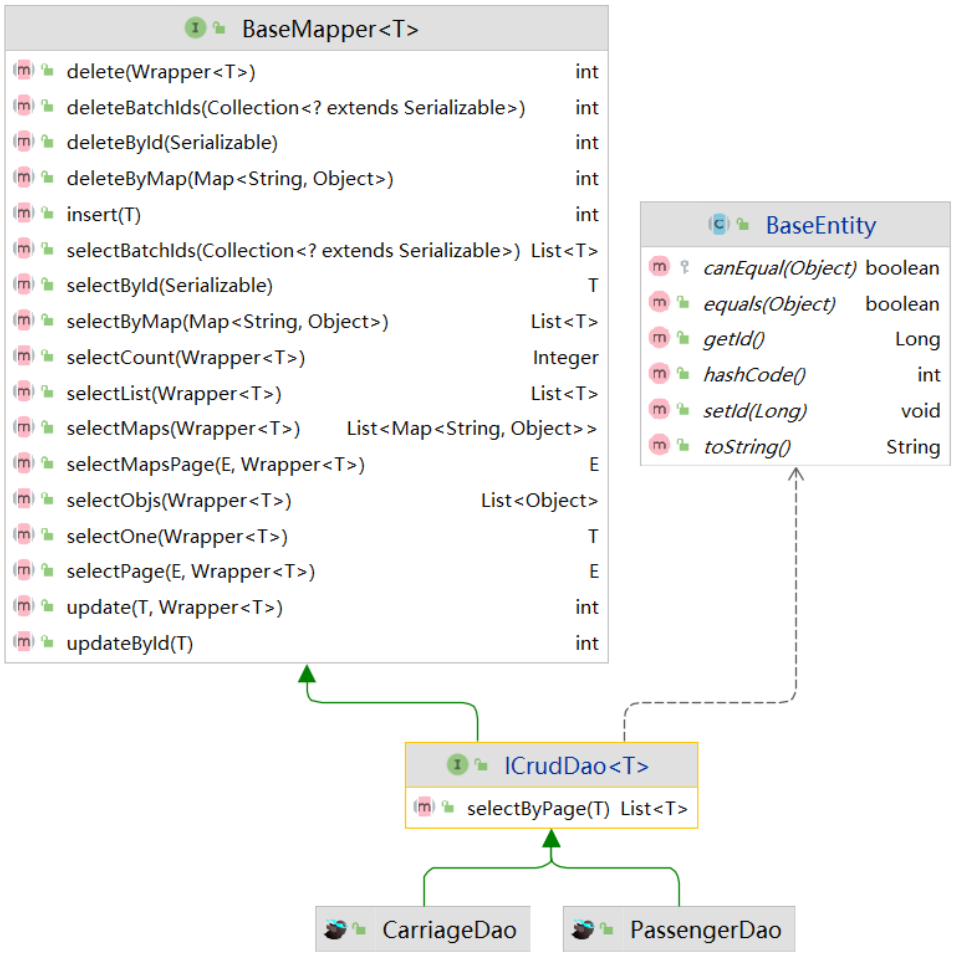

图1–8 Dao基类类图

ICrudDao是dao层基类，Dao层类继承此类，就可以使用Mybatis Plus的BaseMapper提供的标准的增删改查方法了，**注意自定义方法selectByPage，在映射文件中需要实现，用于动态SQL查询**

Java 复制代码

```java

public interface ICrudDao<T extends BaseEntity> extends BaseMapper<T> {

    /**
     * 一般要是用动态sql语句查询
     * @param entity
     * @return
     */
    public List<T> selectByPage(T entity);

}
```

3.1.3 service基类

Service层基类，由接口ICrudService和实现类CrudServiceImpl组成，类图如图1-9所示。

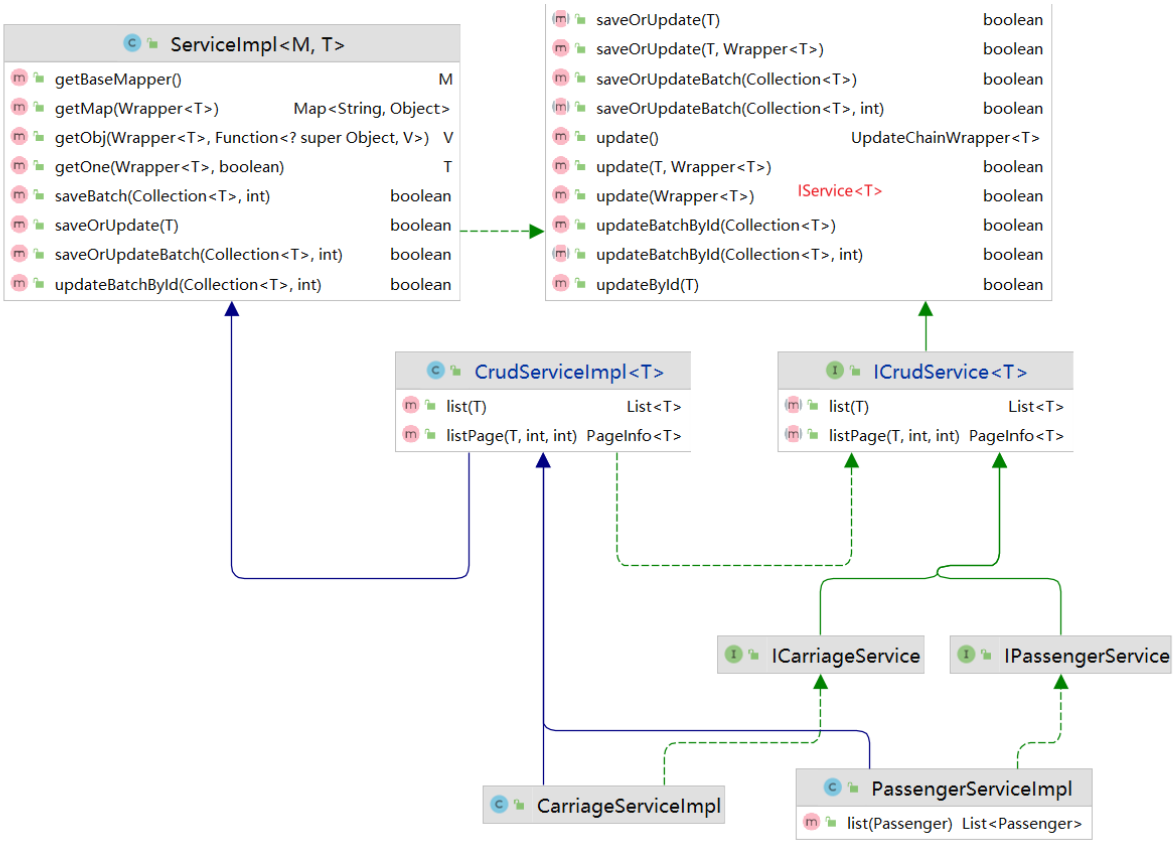

图1-9 Service基类层类图

ICrudService接口源码如下。

Java 复制代码

```java
public interface ICrudService<T extends BaseEntity> extends IService<T> {

    /**
     * 分页查询方法
     * @param entity
     * @param pageNum
     * @param pageSize
     * @return
     */
    PageInfo<T> listPage(T entity, int pageNum, int pageSize);

    /**
     * 查询所有方法
     * @param entity
     * @return
     */
    List<T> list(T entity);

}
```

CrudServiceImpl实现类源码如下。

Java 复制代码

```java
public class CrudServiceImpl<T extends BaseEntity> extends ServiceImpl<ICrudDao<T>, T> implements ICrudService<T> {

    @Override
    public PageInfo<T> listPage(T entity, int pageNum, int pageSize) {
        return PageHelper.startPage(pageNum, pageSize).doSelectPageInfo(() -> {
            baseMapper.selectByPage(entity);
        });
    }

    @Override
    public List<T> list(T entity) {
        return getBaseMapper().selectList(Wrappers.emptyWrapper());
    }

}
```

3.1.4 controler基类

控制层基类类图，如图1–10所示

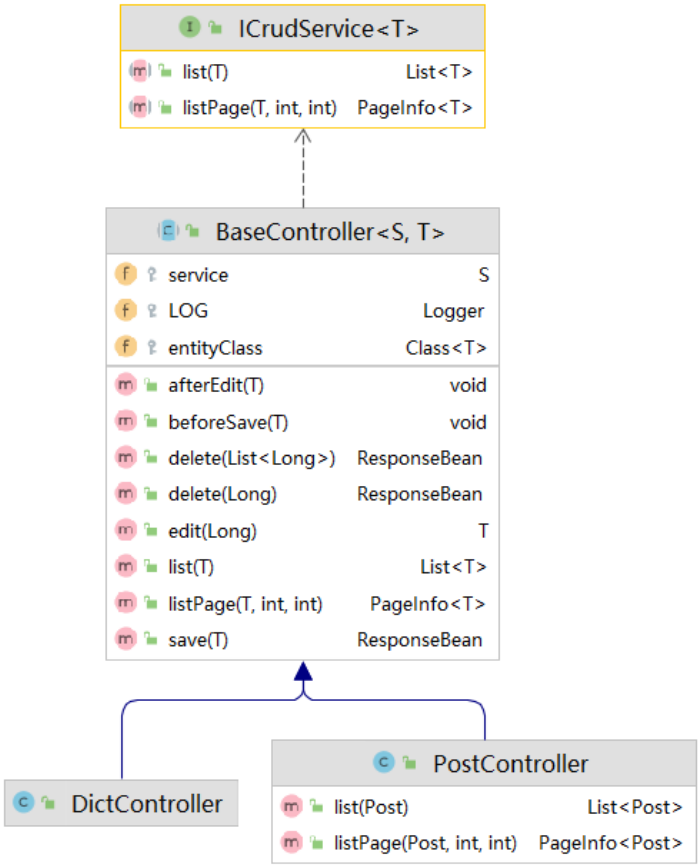

图1–10 控制层基类类图

BaseController源码如下。

Java 复制代码

```java
public abstract class BaseController<S extends ICrudService<T>, T extends BaseEntity> {

    @Autowired
    protected S service;

    protected Logger log = LoggerFactory.getLogger(this.getClass());

    /**
     * 加载
     *
     * @param id
     * @return
     * @throws Exception
     */
    @ApiOperation(value="加载", notes="根据ID加载")
    @GetMapping("/edit/{id}")
    public T edit(@PathVariable Long id) throws Exception {
        T entity = service.getById(id);
        afterEdit(entity);
        return entity;
    }

    /**
     * 分页查询
     * @param entity
     * @param page
     * @param rows
     * @return
     */
    @ApiOperation(value="分页查询", notes="分页查询")
    @PostMapping("/list-page")
    public PageInfo<T> listPage(T entity,
                                @RequestParam(name = "page", defaultValue = "1", required = false) int page,
                                @RequestParam(name = "rows", defaultValue = "10", required = false) int rows) {
        PageInfo<T> result = service.listPage(entity, page, rows);
        return result;
    }

    /**
     * 根据实体条件查询
     * @return
     */
    @ApiOperation(value="查询", notes="根据实体条件查询")
    @RequestMapping(value = "/list", method = {RequestMethod.POST, RequestMethod.GET})
```

```
45      public List<T> list(T entity) {
46          List<T> list = service.list(entity);
47          return list;
48      }
49
50      /**
51       * 增加，修改
52       */
53      @ApiOperation(value="保存", notes="ID存在修改，不存在添加")
54      @PostMapping("/save")
55      public ResponseBean save(T entity) throws Exception {
56          ResponseBean rm = new ResponseBean();
57          try {
58              beforeSave(entity); //保存前处理实体类
59              service.saveOrUpdate(entity);
60              rm.setModel(entity);
61          } catch (Exception e) {
62              e.printStackTrace();
63              rm.setSuccess(false);
64              rm.setMsg("保存失败");
65          }
66          return rm;
67      }
68
69      /**
70       * 删除
71       */
72      @ApiOperation(value="删除", notes="根据ID删除")
73      @GetMapping("/delete/{id}")
74      public ResponseBean delete(@PathVariable Long id) throws Exception {
75          ResponseBean rm = new ResponseBean();
76          try {
77              service.removeById(id);
78              rm.setModel(null);
79          } catch (Exception e) {
80              e.printStackTrace();
81              rm.setSuccess(false);
82              rm.setMsg("保存失败");
83          }
84          return rm;
85      }
86
87      /**
88       * 批量删除
89       */
90      @ApiOperation(value="删除", notes="批量删除")
91      @RequestMapping(value = "/delete", method = {RequestMethod.POST,
    RequestMethod.GET})
92      public ResponseBean delete(@RequestParam List<Long> ids) {
93          ResponseBean rm = new ResponseBean();
```

```
        try {
            service.removeByIds(ids);
        } catch (Exception e) {
            e.printStackTrace();
            rm.setMsg("删除失败");
            rm.setSuccess(false);
        }
        return rm;
    }

    /**
     * 保存前执行
     * @param entity
     * @throws Exception
     */
    public void beforeSave(T entity) throws Exception {
    }

    /**
     * 模板方法：在加载后执行
     * @param entity
     */
    public void afterEdit(T entity) {

    }

}
```

3.2 前端复用

前端的复用思路跟后端一致，通过继承base-list，base-edit组件，把CRUD重复的业务逻辑代码重用到base-list和base-edit组件中实现。

3.2.1 重用逻辑分析

前端用户组件功能展示效果，如图1-11所示。

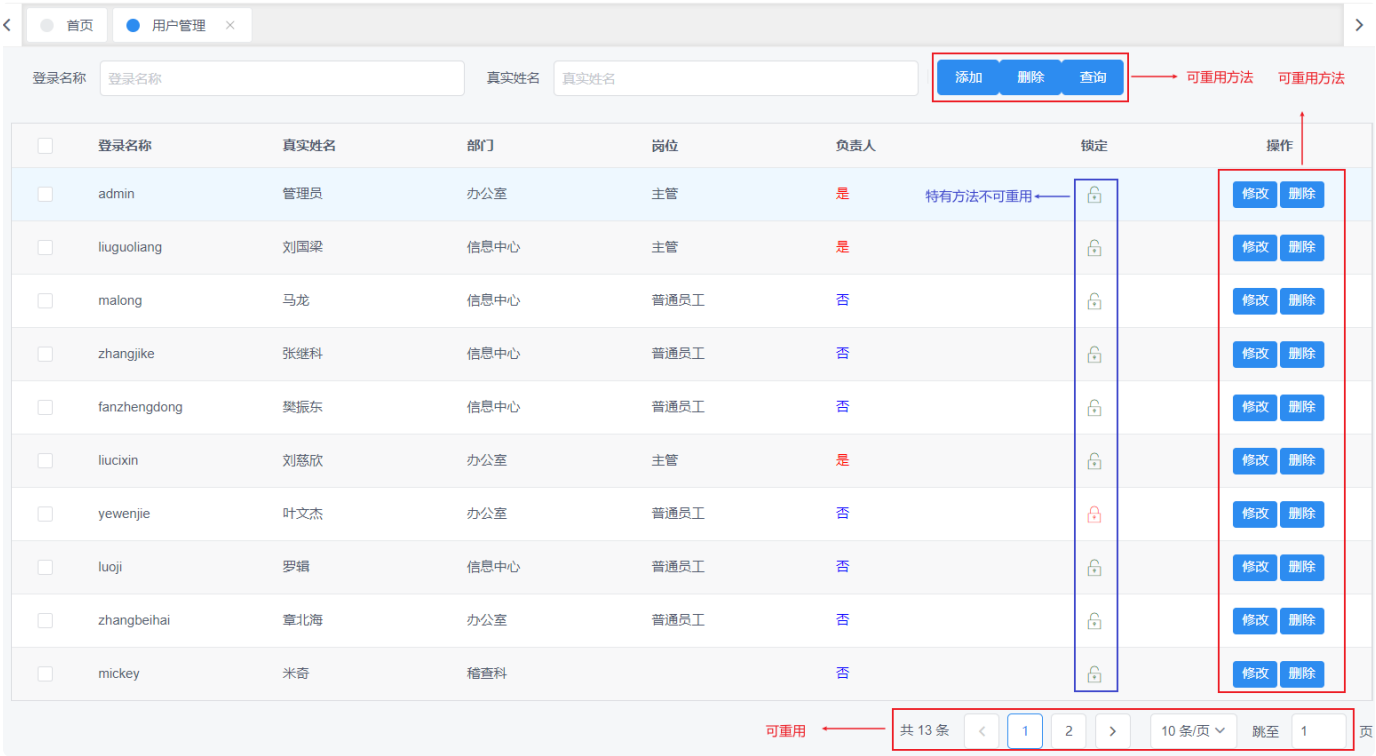

图1–11 用户组件

前端字典组件功能展示效果，如图1–12所示。

图1–12 字典组件

通过分析得到如下结论

- **增删改查功能是乘客和用户都有的功能方法，因此可重用**
- **加入黑名单方法是用户特有方法，乘客没有的方法，因此不可重用**

实现前端组件的重用需要考虑前端组件发送请求给后端的地址，分析如下。

1. 用户

- 分页查询`/security/user/list-page`
- 添加修改保存`/security/user/save`

2. 乘客

- 分页查询`/admin/dict/list-page`
- 添加修改保存`/admin/dict/save`

访问地址的前两部分跟前端路由地址的前两部分相同，如图1-13所示。

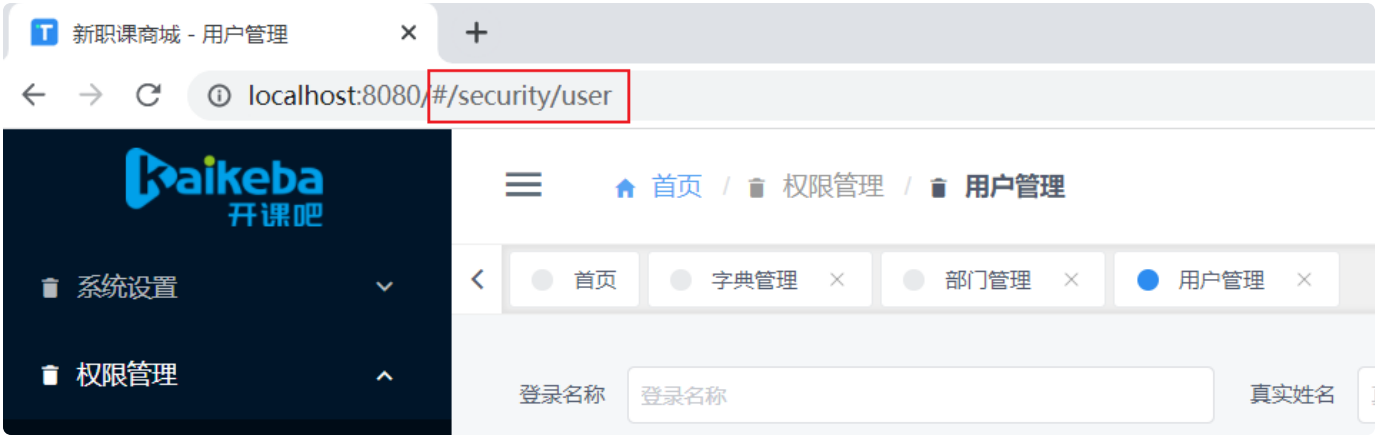

图1-13 用户前端路由地址

同时也要让路由name跟这两个变量产生联系，有此约定，前端组件才能更好的复用

3.2.1 base-list

base-list.js组件实现，列表形态的表格数据CRUD操作，比如用户，乘客，车次等。代码如下。

JavaScript 复制代码

```javascript
import instance from '@/libs/api/index'
import Qs from 'qs'
import { mapMutations } from 'vuex'

export const baseList = {

  data () {
    return {
      // 当前路由的子目录/admin/post/1 -> security
      namespace: '',
      // 当前路由的最后访问路径/admin/post/1-> post
      entityName: '',
      // 初始化信息总条数
      total: 0,
      // 每页显示多少条
      pageSize: 10,
      // 显示的数据
      rows: []
    }
  },

  methods: {
    ...mapMutations([
      'closeTag',
      'addTag'
    ]),
    // 添加
    add () {
      let r = this.$store.state.app.tagNavList.find((item) => {
        return item.name == `edit_${this.namespace}_${this.entityName}`
      })
      if (!r) {
        this.$router.push({
          name: `edit_${this.namespace}_${this.entityName}`
          // query: { id: id }
        })
      } else {
        this.closeTag(r)
        r.query = { id: '' }
        this.$router.push(r)
      }
    },
    // 删除
    remove (id, index) {
      this.$Modal.confirm({
        title: '确认删除',
        content: '确定要删除吗？',
        onOk: () => {
```

```
49          instance.get(`/${this.namespace}/${this.entityName}/delete/` +
    id).then(response => {
50            this.$Message.info('删除成功')
51            this.query()
52          }).catch(error => {
53            console.log(error)
54          })
55        }
56      })
57    },
58    // 批量删除
59    removeBatch () {
60      if (this.$refs.selection.getSelection().length > 0) {
61        this.$Modal.confirm({
62          title: '确认删除',
63          content: '确定要删除吗？',
64          onOk: () => {
65            let params = new URLSearchParams()
66            this.$refs.selection.getSelection().forEach((o) => {
67              params.append('ids', o.id)
68            })
69            instance.post(`/${this.namespace}/${this.entityName}/delete`,
    params).then(response => {
70              this.$Message.info('删除成功')
71              this.query()
72            })
73          }
74        })
75      } else {
76        this.$Message.info('请选择删除的数据')
77      }
78    },
79    // 修改
80    edit (id) {
81      let r = this.$store.state.app.tagNavList.find((item) => {
82        return item.name == `edit_${this.namespace}_${this.entityName}`
83      })
84      if (!r) {
85        this.$router.push({
86          name: `edit_${this.namespace}_${this.entityName}`,
87          query: { id: id }
88        })
89      } else {
90        this.closeTag(r)
91        r.query = { id: id }
92        this.$router.push(r)
93      }
94    },
95    // 查询
96    query () {
```

```
97        instance.post(`/${this.namespace}/${this.entityName}/list-page`,
    Qs.stringify(this.formData)).then(response => {
98          this.rows = response.data.list
99          this.total = response.data.total
100       }).catch(error => {
101         console.log(error)
102       })
103     },
104     // 分页
105     changePage (index) {
106       this.formData.page = index
107       this.query()
108     },
109     // 设置每页行数
110     changePageSize (size) {
111       this.formData.page = 1
112       this.formData.rows = size
113       this.query()
114     }
115   },
116   mounted () {
117     let arrays = this.$route.path.split('/')
118     this.namespace = arrays[1]
119     this.entityName = arrays[2]
120     this.query()
121   }
122 }
```

注意代码中的2个关键

- namespace：对应模块名
- entityName：对应模块中管理的实体名称

通过上面代码得出，这两个变量对用路由中的第二个和第三个子目录，用户路由为`/admin/user`，得到

- namespace=admin
- entityName=user

组件中的所有id和方法调用都与这两个变量有关，以及访问路径都跟这两个变量有关系，我们做好约定，才能进行代码复用，这里要特别注意，具体使用时直接混入即可代码，如图1–14所示。

```
base-list.js   list.vue
37  import {baseList} from '@/libs/crud/base-list'
38
39  export default {
40    mixins: [baseList],
41    data () {
42      return {
```

图1-14 复用前端base-list组件

3.2.2 base-edit

base-edit.js组件实现了，前端修改，添加业务代码的复用，实现思路跟base-list相似，比如用户修改，乘客修改等。代码如下。

JavaScript 复制代码

```javascript
import instance from '@/libs/api/index'
import Qs from 'qs'
import { mapMutations } from 'vuex'

export const baseEdit = {
  data () {
    return {
      // 当前路由的子目录/admin/post/1 -> security
      namespace: '',
      // 当前路由的最后访问路径/admin/post/1-> post
      entityName: ''
    }
  },
  methods: {
    ...mapMutations([
      'closeTag'
    ]),
    /**
         * 模板方法：提交前用来处理保存的数据
         */
    beforeSubmit () {
      alert('b')
    },

    // 提交
    handleSubmit (name) {
      this.$refs[name].validate((valid) => {
        if (valid) {
          instance.post(`/${this.namespace}/${this.entityName}/save`, Qs.stringify(this.formData, { arrayFormat: 'repeat' })).then(response => {
            this.$Message.success(response.data.msg)
            this.go2list()
          })
        } else {
          this.$Message.error('Fail!')
        }
      })
    },
    // 重置
    handleReset (name) {
      this.$refs[name].resetFields()
    },
    // 根据ID加载数据
    get (id) {
      instance.get(`/${this.namespace}/${this.entityName}/edit/` + id).then(response => {
        this.formData = Object.assign(response.data)
      }).catch(error => {
```

```
47          console.log(error)
48        })
49      },
50
51      go2list () {
52        this.closeTag(this.$route)
53      }
54    },
55
56    mounted () {
57      let arrays = this.$route.name.split('_')
58      this.namespace = arrays[1]
59      this.entityName = arrays[2]
60
61      let id = this.$route.query.id
62      if (id) {
63        this.get(id)
64      } else {
65        // Object.keys(this.formData).forEach(key => this.formData[key] = '')
66      }
67    }
68  }
```

3.2.3 base-tree

树状结构数据的重用逻辑和表格数据相似，无非就是加入父节点ID（parentId）处理，比如下图部门和菜单组件的功能，如图1-15所示

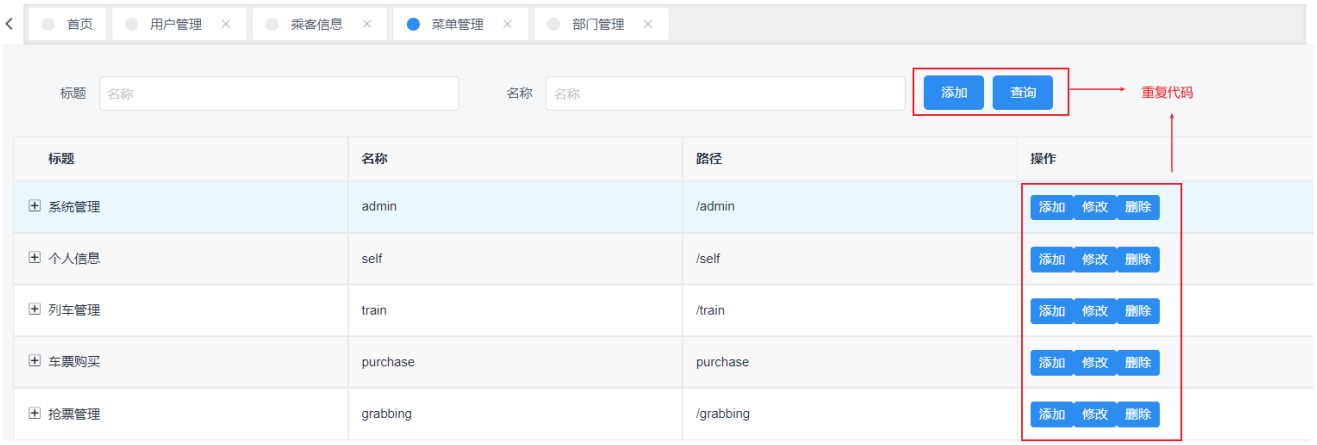

图1-15 树状结构组件

base-tree-list.js实现了，前端树状结构数据列表的业务代码复用，比如部门和菜单列表操作。代码如下。

JavaScript 复制代码

```javascript
import Qs from 'qs'
import Vue from 'vue'
import ZkTable from 'vue-table-with-tree-grid'
import { listToTree } from '@/libs/util'
import instance from '@/libs/api/index'


import {baseList} from './base-list'
Vue.use(ZkTable)

export const baseTreeList = {
  mixins: [baseList],
  // 表格各行的数据
  data () {
    return {
      props: {
        stripe: true, // 是否显示间隔斑马纹
        border: true, // 是否显示纵向边框
        showHeader: true, // 是否显示表头
        showSummary: false, // 是否显示表尾合计行
        showRowHover: true, // 鼠标悬停时，是否高亮当前行
        showIndex: false, // 是否显示数据索引
        treeType: true, // 是否为树形表格
        isFold: true, // 树形表格中父级是否默认折叠
        expandType: false, // 是否为展开行类型表格（为 True 时，需要添加作用域插槽，它可以获取到 row, rowIndex）
        selectionType: false// 是否显示间隔斑马纹
      },
      // 初始化信息总条数
      total: 0,
      // 每页显示多少条
      pageSize: 10,
      // 显示的数据
      rows: []
    }
  },
  methods: {
    // 查询
    query () {
      instance.post(`/${this.namespace}/${this.entityName}/list`, Qs.stringify(this.formData)).then(response => {
        this.rows = listToTree(response.data)
      }).catch(error => {
        console.log(error)
      })
    },
    addRoot () {
      this.$router.push({
```

```
47          name: `edit_${this.namespace}_${this.entityName}`
48        })
49      },
50      // 添加
51      addChild (id) {
52        this.$router.push({
53          name: `edit_${this.namespace}_${this.entityName}`,
54          query: {parentId: id}
55        })
56      }
57    }
58  }
```

base-tree-edit组件实现了，树状结构数据的修改组件的复用，也就是增加了parentId的处理，代码如下。

JavaScript 复制代码

```
1  import { baseEdit } from './base-edit'
2
3  export const baseTreeEdit = {
4    mixins: [baseEdit],
5    created () {
6      let arrays = this.$route.name.split('_')
7      this.namespace = arrays[1]
8      this.entityName = arrays[2]
9
10     let id = this.$route.query.id
11     this.formData.parentId = this.$route.query.parentId
12     if (id) {
13       this.get(id)
14     }
15   }
16 }
```

4. 岗位管理

4.1 后端代码

4.1.1 实体类

Java 复制代码

```java
package com.lxs.legou.admin.po;

import com.baomidou.mybatisplus.annotation.TableField;
import com.baomidou.mybatisplus.annotation.TableName;
import com.lxs.legou.core.po.BaseEntity;
import lombok.Data;

@Data
@TableName("post_")
public class Post extends BaseEntity {

  @TableField("name_")
  private String name;
  @TableField("title_")
  private String title;
  @TableField("desc_")
  private String desc;

}
```

4.1.2 dao

Java 复制代码

```java
package com.lxs.legou.admin.dao;

import com.lxs.legou.admin.po.Post;
import com.lxs.legou.core.dao.ICrudDao;

public interface PostDao extends ICrudDao<Post> {

}
```

映射文件，主要为了实现动态查询，代码如下。

Plain Text 复制代码

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.lxs.legou.admin.dao.PostDao">

  <select id="selectByPage" resultType="Post">
    select
      *
    from
      post_
    <where>
    <if test="title != null and title != ''">
      title_ like '%${title}%'
    </if>
    <if test="name != null and name != ''">
      and name_ like '%${name}%'
    </if>
    </where>
  </select>

</mapper>
```

4.1.3 service

接口

Java 复制代码

```java
package com.lxs.legou.admin.service;

import com.lxs.legou.admin.po.Post;
import com.lxs.legou.core.service.ICrudService;

/**
 * @Title:
 * @Description:
 *
 * @Copyright 2019 lxs - Powered By 雪松
 * @Author: lxs
 * @Date:  2019/10/9
 * @Version V1.0
 */
public interface IPostService extends ICrudService<Post> {

}
```

实现类

Java 复制代码

```java
package com.lxs.legou.admin.service.impl;

import com.lxs.legou.admin.po.Post;
import com.lxs.legou.admin.service.IPostService;
import com.lxs.legou.core.service.impl.CrudServiceImpl;
import org.springframework.stereotype.Service;

@Service
public class PostServiceImpl extends CrudServiceImpl<Post> implements IPostService {

}
```

4.1.4 Mybatis配置类

Java 复制代码

```java
@Configuration
@MapperScan("com.**.dao")
public class MybatisPlusConfig {

    /**
     * 分页插件
     */
    @Bean
    public PaginationInterceptor paginationInterceptor() {
        // 开启 count 的 join 优化,只针对 left join !!!
        return new PaginationInterceptor().setCountSqlParser(new JsqlParserCountOptimize(true));
    }

    @Bean
    public PageInterceptor pageInterceptor() {
        return new PageInterceptor();
    }
}
```

4.2 前端组件

4.2.1 list.vue

JavaScript 复制代码

```
<template>
    <div>
        <div>
            <Form ref="formData" :model="formData" :label-width="80">
                <Row style="margin-top: 10px;">
                    <Col span="8">
                    <FormItem label="名称" prop="name">
                        <Input v-model="formData.name" placeholder="名称">
</Input>
                    </FormItem>
                    </Col>
                    <Col span="8">
                    <FormItem label="描述" prop="title">
                        <Input v-model="formData.title" placeholder="描述">
</Input>
                    </FormItem>
                    </Col>
                    <Col span="8">
                        <Divider type="vertical" />
                        <Button type="primary" @click="add">添加</Button>
                        <Button type="primary" @click="removeBatch"
style="margin-left: 8px">删除</Button>
                        <Button type="primary" @click="query" style="margin-
left: 8px">查询</Button>
                    </Col>
                </Row>
            </Form>
        </div>

        <div>
            <Table stripe ref="selection" :columns="columns" :data="rows">
</Table>
        </div>
        <div class="paging">
            <Page :total="total" :page-size="pageSize" show-sizer show-
elevator show-total
                  @on-change="changePage" @on-page-size-
change="changePageSize"></Page>
        </div>
    </div>
</template>
<style scoped>
    .paging {
        float: right;
        margin-top: 10px;
    }
</style>
<script>
```

```
42 import {baseList} from '@/libs/crud/base-list'
43
44 export default {
45   mixins: [baseList],
46   data () {
47     return {
48       formData: {
49         name: '',
50         title: ''
51       },
52       columns: [
53         {
54           type: 'selection',
55           width: 60,
56           align: 'center'
57         },
58         {
59           title: '名称',
60           key: 'name'
61         },
62         {
63           title: '描述',
64           key: 'title'
65         },
66         {
67           title: '操作',
68           key: 'action',
69           width: 150,
70           align: 'center',
71           render: (h, params) => {
72             return h('div', [
73               h('Button', {
74                 props: {
75                   type: 'primary',
76                   size: 'small'
77                 },
78                 style: {
79                   marginRight: '5px'
80                 },
81                 on: {
82                   click: () => {
83                     this.edit(params.row.id)
84                   }
85                 }
86               }, '修改'),
87               h('Button', {
88                 props: {
89                   type: 'primary',
90                   size: 'small'
91                 },
```

```
92                  on: {
93                    click: () => {
94                      this.remove(params.row.id, params.index)
95                    }
96                  }
97                }, '删除')
98              ])
99            }
100         }
101       ]
102     }
103   }
104 }
105 </script>
```

4.2.2 edit.vue

JavaScript 复制代码

```

<template>

    <Form ref="form" :model="formData" :rules="ruleValidate" :label-width="80">
        <input type="hidden" v-model="formData.id"/>

        <FormItem label="名称" prop="name">
            <Input v-model="formData.name"></Input>
        </FormItem>
        <FormItem label="描述" prop="title">
            <Input v-model="formData.title"></Input>
        </FormItem>

        <FormItem label="备注" prop="desc">
            <Input type="textarea" :rows="10" v-model="formData.desc" ></Input>
        </FormItem>

        <FormItem>
            <Button type="primary" @click="handleSubmit('form')">保存</Button>
            <Button type="primary" @click="go2list()" style="margin-left:8px">关闭</Button>
        </FormItem>

    </Form>

</template>

<script>
import {baseEdit} from '@/libs/crud/base-edit'

export default {
  mixins: [baseEdit],
  data () {
    return {
      formData: {
        id: '',
        name: '',
        title: '',
        desc: ''
      },
      ruleValidate: {
        name: [
          {required: true, message: '名称不能为空', trigger: 'blur'}
        ]
      }
    }
```

```
  }
}
</script>
```

5. 部门管理

5.1 后端代码

5.1.1 实体类

Java 复制代码

```java
package com.lxs.legou.admin.po;

import com.baomidou.mybatisplus.annotation.TableField;
import com.baomidou.mybatisplus.annotation.TableName;
import com.lxs.legou.core.po.BaseTreeEntity;
import lombok.Data;

/**
 * @Title:
 * @Description:
 *
 * @Copyright 2019 lxs - Powered By 雪松
 * @Author: lxs
 * @Date:  2019/10/9
 * @Version V1.0
 */
@Data
@TableName("dept_")
public class Dept extends BaseTreeEntity {

  @TableField("address_")
  private String address;
  @TableField("tel_")
  private String tel;
  @TableField("desc_")
  private String desc;

  public String getLabel() { //treeselect需要的属性
    return this.getTitle();
  }

}
```

5.1.2 dao

Java 复制代码

```java
package com.lxs.legou.admin.dao;

import com.lxs.legou.admin.po.Dept;
import com.lxs.legou.core.dao.ICrudDao;

/**
 * @file DeptDao.java
 * @Copyright (C) http://www.lxs.com
 * @author lxs
 * @email lxosng77@163.com
 * @date 2018/7/14
 */
public interface DeptDao extends ICrudDao<Dept> {

}
```

5.1.3 service

接口

Java 复制代码

```java
package com.lxs.legou.admin.service;

import com.lxs.legou.admin.po.Dept;
import com.lxs.legou.core.service.ICrudService;
import org.springframework.stereotype.Service;

/**
 * @Title:
 * @Description:
 *
 * @Copyright 2019 lxs - Powered By 雪松
 * @Author: lxs
 * @Date:  2019/10/9
 * @Version V1.0
 */
@Service
public interface IDeptService extends ICrudService<Dept> {

}
```

实现类

Java 复制代码

```java
package com.lxs.legou.admin.service.impl;

import com.baomidou.mybatisplus.core.conditions.query.QueryWrapper;
import com.baomidou.mybatisplus.core.toolkit.Wrappers;
import com.lxs.legou.admin.po.Dept;
import com.lxs.legou.admin.service.IDeptService;
import com.lxs.legou.core.service.impl.CrudServiceImpl;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
public class DeptServiceImpl extends CrudServiceImpl<Dept> implements IDeptService {

    public List<Dept> list(Dept entity) {
        QueryWrapper<Dept> queryWrapper = Wrappers.<Dept>query();
        if (null != entity.getAddress() && !"".equals(entity.getAddress().trim())) {
            queryWrapper.like("address", entity.getAddress());
        }
        if (null != entity.getTitle() && !"".equals(entity.getTitle().trim())) {
            queryWrapper.like("title", entity.getTitle());
        }
        return baseMapper.selectList(queryWrapper);
    }
}
```

5.1.4 Controller

Java 复制代码

```java
package com.lxs.legou.admin.controller;

import com.lxs.legou.admin.po.Dept;
import com.lxs.legou.admin.service.IDeptService;
import com.lxs.legou.core.controller.BaseController;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;


@RestController
@RequestMapping("/dept")
public class DeptController extends BaseController<IDeptService, Dept> {

}
```

5.2 前端组件

5.2.1 list.vue

JavaScript 复制代码

```
<template>
    <div>
        <div>
            <Form ref="formData" :model="formData" :label-width="80">
                <Row style="margin-top: 10px;">
                    <Col span="8">
                    <FormItem label="名称" prop="title">
                        <Input v-model="formData.title" placeholder="名称">
</Input>
                    </FormItem>
                    </Col>
                    <Col span="8">
                    <FormItem label="地址" prop="address">
                        <Input v-model="formData.address" placeholder="地址">
</Input>
                    </FormItem>
                    </Col>
                    <Col span="8">
                        <Divider type="vertical" />
                        <Button type="primary" @click="addRoot">添加</Button>
                        <Button type="primary" @click="query" style="margin-
left: 8px">查询</Button>
                    </Col>
                </Row>
            </Form>
        </div>

        <div>
            <zk-table
                    ref="table"
                    sum-text="sum"
                    index-text="#"
                    :data="rows"
                    :columns="columns"
                    :stripe="props.stripe"
                    :border="props.border"
                    :show-header="props.showHeader"
                    :show-summary="props.showSummary"
                    :show-row-hover="props.showRowHover"
                    :show-index="props.showIndex"
                    :tree-type="props.treeType"
                    :is-fold="props.isFold"
                    :expand-type="props.expandType"
                    :selection-type="props.selectionType">
                <!-- 原文中 scope="scope" 会警告， 2.5版本后应为slot-
scope="scope"-->
                <template slot="action" slot-scope="scope">
                    <Button type="primary" size="small"
```

```
@click="addChild(scope.row.id)">添加</Button>
                    <Button type="primary" size="small" @click="edit(scope.row.id)" style="margin-left: 2px">修改</Button>
                    <Button type="primary" size="small" @click="remove(scope.row.id)" style="margin-left: 2px">删除</Button>
                </template>
            </zk-table>
        </div>
    </div>
</template>

<style scoped>
    .paging {
        float: right;
        margin-top: 10px;
    }
</style>

<script>
import {baseTreeList} from '@/libs/crud/base-tree-list'

export default {
  mixins: [baseTreeList],
  // 表格各行的数据
  data () {
    return {
      formData: {
        title: '',
        address: ''
      },
      // 表格标题数据
      columns: [
        {
          label: '名称',
          prop: 'title'
        },
        {
          label: '地址',
          prop: 'address'
        },
        {
          label: '电话',
          prop: 'tel'
        },
        {
          label: '操作',
          prop: 'action',
          type: 'template',
          template: 'action'
        }
```

```
92          ]
93        }
94      }
95    }
96    </script>
```

5.2.2 edit.vue

JavaScript 复制代码

```
<template>

    <Form ref="form" :model="formData" :rules="ruleValidate" :label-width="80">
        <input type="hidden" v-model="formData.id"/>
        <input type="hidden" v-model="formData.parentId"/>

        <Row>
            <Col span="12">
            <FormItem label="名称" prop="title">
                <Input v-model="formData.title"></Input>
            </FormItem>
            </Col>
            <Col span="12">
            <FormItem label="排序" prop="order">
                <Input v-model="formData.order"></Input>
            </FormItem>
            </Col>
        </Row>
        <Row>
            <Col span="12">
            <FormItem label="电话" prop="tel">
                <Input v-model="formData.tel"></Input>
            </FormItem>
            </Col>
            <Col span="12">
            <FormItem label="是否展开" prop="expand">
                <Checkbox v-model="formData.expand"></Checkbox>
            </FormItem>
            </Col>
        </Row>

        <FormItem label="地址" prop="address">
            <Input v-model="formData.address"></Input>
        </FormItem>

        <FormItem label="描述" prop="desc">
            <Input type="textarea" :rows="10" v-model="formData.desc" ></Input>
        </FormItem>

        <FormItem>
            <Button type="primary" @click="handleSubmit('form')">保存</Button>
            <Button type="primary" @click="go2list()" style="margin-left:8px">关闭</Button>
        </FormItem>

    </Form>
```

```
46
47  </template>
48
49  <script>
50
51  //import {baseEdit} from '@/libs/crud/base-edit'
52  import {baseTreeEdit} from '@/libs/crud/base-tree-edit'
53  import {validateTel} from '@/libs/validate/base-rule.js'
54
55  export default {
56    mixins: [baseTreeEdit],
57    data () {
58      return {
59        formData: {
60          id: '',
61          parentId: '',
62          title: '',
63          order: '',
64          tel: '',
65          address: '',
66          desc: '',
67          expand: false
68        },
69        ruleValidate: {
70          title: [
71            {required: true, message: '名称不能为空', trigger: 'blur'}
72          ],
73          tel: [
74            {required: true, message: '电话不能为空', trigger: 'blur'},
75            { validator: validateTel, trigger: 'blur' }
76          ]
77        }
78      }
79    }
80  }
81  </script>
```