**数据库课程应用设计**

**超市会员管理系统项目报告**

目录

[1. 绪论 3](#_Toc6561601)

[2. 相关技术介绍 4](#_Toc6561602)

[2.1. 后端 4](#_Toc6561603)

[2.1.1. 持久化技术-ORM框架Hibernate 4](#_Toc6561604)

[2.1.2. Java分层开发框架-Spring 4](#_Toc6561605)

[2.1.3. 控制器框架-Struts2 4](#_Toc6561606)

[2.1.4. 其它 4](#_Toc6561607)

[2.2. 前端 4](#_Toc6561608)

[2.2.1. Javascript框架 4](#_Toc6561609)

[2.2.2. CSS框架 5](#_Toc6561610)

[3. 需求分析 5](#_Toc6561611)

[3.1. 目标 5](#_Toc6561612)

[3.2. 用户特点 5](#_Toc6561613)

[3.3. 需求规定 6](#_Toc6561614)

[3.3.1. 功能需求规定 6](#_Toc6561615)

[3.3.2. 性能需求规定 7](#_Toc6561616)

[3.3.3. 输入输出要求 7](#_Toc6561617)

[4. 概要设计 8](#_Toc6561618)

[4.1. 总体结构设计 8](#_Toc6561619)

[4.2. 系统功能模块设计 8](#_Toc6561620)

[4.3. 业务流程设计 9](#_Toc6561621)

[4.4. 用户界面设计 9](#_Toc6561622)

[4.5. 持久化数据结构设计(ER图) 10](#_Toc6561623)

[5. 详细设计 10](#_Toc6561624)

[5.1. 功能模块详细设计 10](#_Toc6561625)

[5.1.1. 登录 10](#_Toc6561626)

[5.1.2. 会员信息查询 12](#_Toc6561627)

[5.1.3. 修改会员信息 13](#_Toc6561628)

[5.1.4. 删除会员信息 15](#_Toc6561629)

[5.1.5. 注册新会员 16](#_Toc6561630)

[5.1.6. 筛选消费记录 18](#_Toc6561631)

[5.2. 逻辑表结构设计 19](#_Toc6561632)

[6. 系统实现 20](#_Toc6561633)

[6.1. 配置说明 20](#_Toc6561634)

[6.2. Spring配置-实现控制反转 20](#_Toc6561635)

[6.3. model层实现-ORM映射 21](#_Toc6561636)

[6.4. dao层实现-封装sql查询 21](#_Toc6561637)

[6.5. Service层-核心业务逻辑 21](#_Toc6561638)

[6.6. Action层-通过Struts控制业务流程 22](#_Toc6561639)

[6.7. DTO-数据承载模型 23](#_Toc6561640)

[6.8. 基于Vue.js的前端模块化实现 23](#_Toc6561641)

[7. 测试 24](#_Toc6561642)

[7.1.1. DAO层接口测试 24](#_Toc6561643)

[7.1.2. Service层接口测试 25](#_Toc6561644)

[8. 使用说明 25](#_Toc6561645)

[8.1. 登录 25](#_Toc6561646)

[8.2. 管理系统 26](#_Toc6561647)

[8.2.1. 会员信息管理 26](#_Toc6561648)

[8.2.2. 消费记录查询 28](#_Toc6561649)

# 绪论

本课设基于模拟场景的大致需求为：设计一个面向超市特定员工的，提供会员注册、会员个人信息记录、会员管理、积分计算等功能的系统。该项目主要的实现目标为:搭建一套完整的可维护可拓展的前后端运行框架、完成基本需求功能。

本报告主体由相关技术介绍、需求分析、概要设计、详细设计、系统实现等多个部分组成。内容概要如下:

相关技术介绍简要介绍了本项目计划于实现中使用的框架与技术；

需求分析罗列了项目计划要实现的功能以及对功能的特殊要求；

概要设计通过若干张图表介绍了项目的主体框架构成思路、主要业务功能及其流程与数据库概念结构设计。

详细设计按概要设计中的功能模块图，将系统分为若干功能，通过UML类图与时序图描述每个功能模块的内部设计、各个子模块之间的依赖以及该功能模块运行时的具有运行流程。

系统实现部分主要介绍了项目与相关技术的关联，介绍了如何利用各种框架技术的功能特性去实现特定的需求。

测试部分简要介绍了底层接口的单元测试过程。

# 相关技术介绍

## 后端

### 持久化技术-ORM框架Hibernate

ORM(对象关系映射)是严格按照OOP的模式封装传统JDBC的一种数据库连接技术。

### Java分层开发框架-Spring

#### 控制反转(IoC)

由Spring框架控制项目各组件的依赖关系，保证项目各层模块之间的松耦合。

#### 面向切面(AOP)

通过Spring框架提供的AOP支持，对项目各不同组件的使用统一的异常捕获、错误输出。

### 控制器框架-Struts2

拦截/响应请求，控制各Action组件执行操作、发送/接受数据。

### 其它

* 通用java日志工具-log4j。
* 通用java单元测试工具-Junit。
* 微软Sql server JDBC驱动。

## 前端

### Javascript框架

#### MVVM式前端框架-Vue.js

通过数据绑定将前端视图UI与前端业务逻辑分离解耦。

Vue还提供了一种较为合理的模式去模块化前端业务代码。

#### HTTP异步请求(AJAX)库-Axios

基于ES6的Promise特性，可避免大量的回调函数嵌套。

### CSS框架

#### 矢量图标库-fontawesome

流行的矢量图标库、可根据字体设置自适应的css框架。

#### 轻量级css样式库-Bulma

纯css代码实现的框架，不像bootstrap依赖于jquery，回避与vue.js的兼容性问题。

# 需求分析

## 目标

功能完善，界面简洁，结构完整的超市会员管理系统。

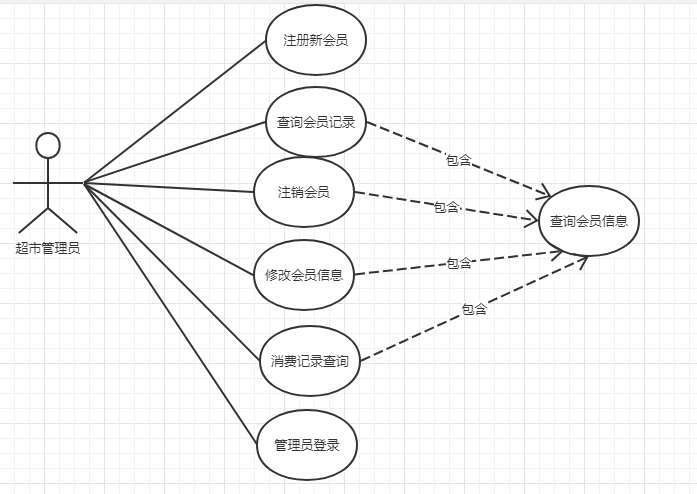
超市后台管理系统仅后台管理者使用，方便后台管理者对超市会员的信息管理和处理。

## 用户特点

本系统的使用者为超市招聘的后台管理人员，使用频率为每天。

## 需求规定

### 功能需求规定



#### 超市管理人员登录

防止非超市员工的游客访问网站，登录系统的账号密码由网站维护人员在后台添加。

#### 注册新会员

注册新会员由超市员工操作，输入会员需要的所有信息，包括会员年龄，会员性别，会员姓名，会员手机号。

然后自动生成会员ID和注册会员日期并初始化会员积分。

#### 查询会员信息

可通过会员任意信息条目搜索相应的会员记录

#### 删除会员信息

可以选择删除会员信息，在删除之前需要有是否确认删除的弹窗，避免失误操作。

#### 会员信息修改

进如某位会员的详细信息界面后，可以选择修改这位会员的会员信息，允许修改会员年龄，会员性别，会员姓名，会员手机号。

#### 消费记录查询

员工可以通过消费记录任意信息条目筛选消费记录

#### 特殊需求

会员积分由会员消费记录产生的积分变动计算而来。

消费记录在本会员管理系统中无法。

### 性能需求规定

#### 时间特性要求

1. 各功能组件响应数据到UI界面的时间要求在3000ms内
2. UI界面提交数据到后端至到后端响应时间要求要求在5000ms内

#### 灵活性

当需求发生某些变化时，该软件对以下变换具备适应能力：

1. 窗口尺寸的变化；
2. 运行环境的变化；
3. 底层数据库接口的变化；

### **输入输出要求**

需要检查输入的功能有会员查询、会员注册和会员修改。

各数据输入要求如下：

电话号码只能输入11位合法存在的手机号码；

姓名只允许输入中文和英文字符；

年龄只允许输入数字；

性别只能输入男或者女。

# 概要设计

## 总体结构设计

上图为项目大致的分层结构，以及引入的框架大致作用的层级。

本项目主体前后端分离的模式，前后端通过ajax技术异步通信。

前端大致上将设计为一SPA(单页面应用程序)，页面功能由各种模块(基本上由一对View-ViewModel)组成，各模块在交互过程中由前端脚本语言动态加载。

后端大致以Spring的MVC分层思想为主

DTO(data to object)层是一组在POJO的基础上扩展的模型对象，主要负责在Action层收发数据前的序列化/反序列化。

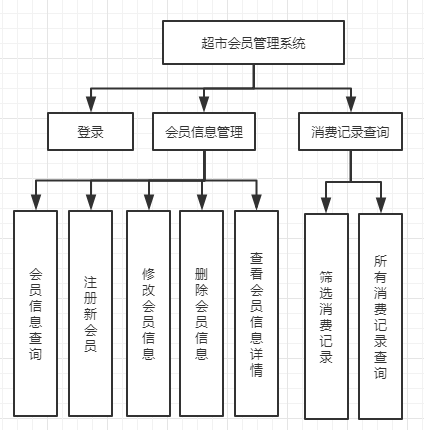
Action层为控制器层，负责处理/响应请求，通过DTO层收发数据，通过Service层处理数据。

Service为核心业务逻辑层，通过DAO层读写数据，向Action层提供服务。

DAO层封装所有数据库操作，向Service提供一组读写数据库的接口。

Model层是一组POJO，直接由ORM引入的框架映射于DB中的表项

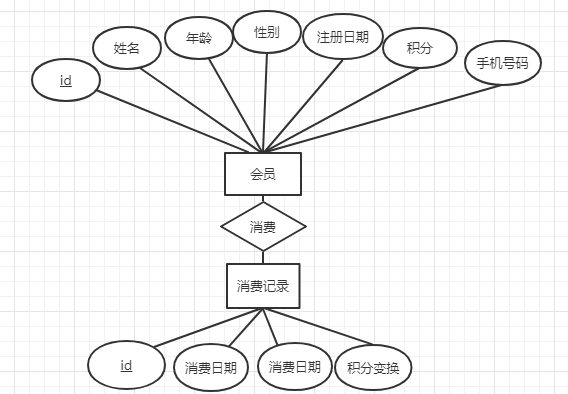
## 系统功能模块设计



## 业务流程设计

## 用户界面设计

## 持久化数据结构设计(ER图)



# 详细设计

## 功能模块详细设计

后端面向前端的提供功能的各个接口由各种功能模块组提供。

### 登录

#### 程序描述

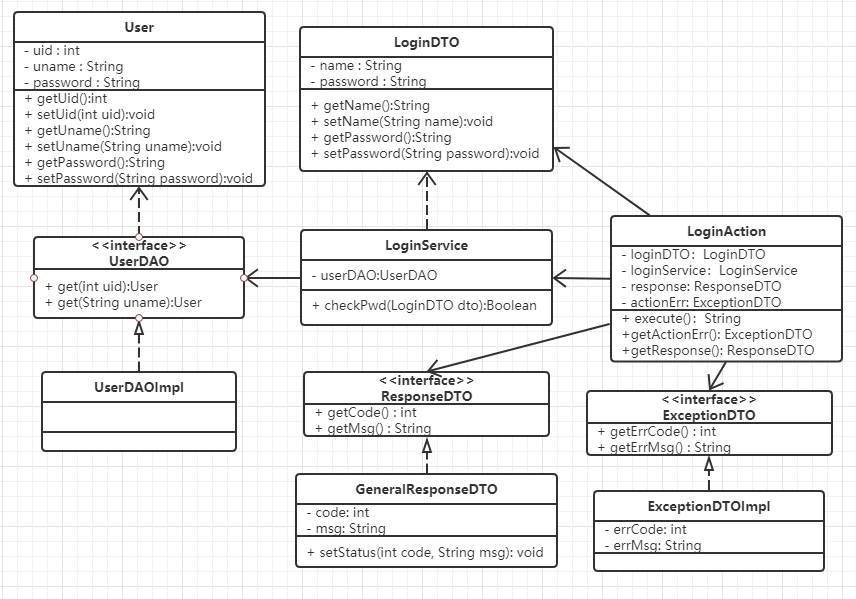
防止非超市员工的游客访问网站，登录系统的账号密码由网站维护人员在后台添加。

#### 输入与输出

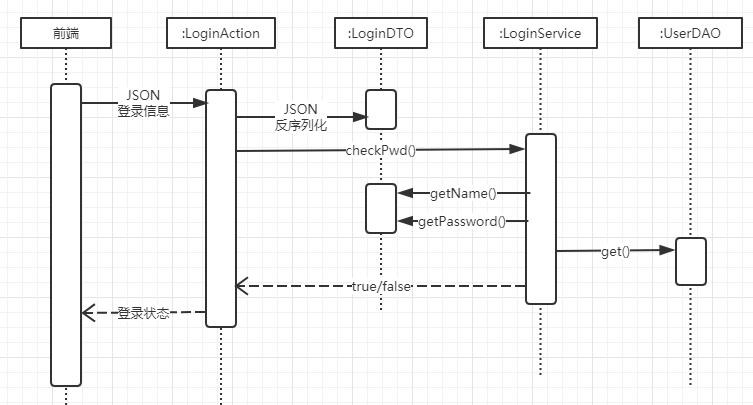
输入：JSON数据{user：[值],password：[值]}；

输出：JSON数据{code：[值],msg：[值]}；

#### 模块内部结构设计



#### 流程逻辑设计



### 会员信息查询

#### 程序描述

可通过会员任意信息条目搜索相应的会员记录

#### 输入与输出

输入：JSON对象数据{key：[key值],[key值]：[条目值]}；

输出：JSON数组数据[{id: [条目值],

name: [条目值],

age: [条目值],

sex: [条目值],

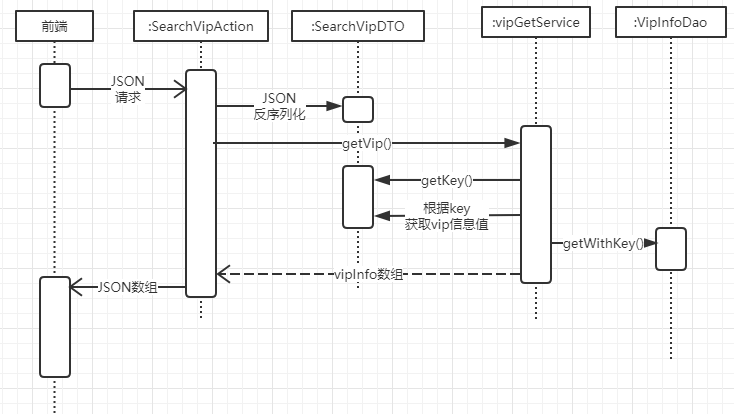
regDate: [条目值],

phone: [条目值],

points: [条目值]},..]；

#### 模块内部结构设计

#### 流程逻辑设计



### 修改会员信息

#### 程序描述

进如某位会员的详细信息界面后，可以选择修改这位会员的会员信息，允许修改会员年龄，会员性别，会员姓名，会员手机号。

#### 输入与输出

输入：JSON对象{id: [条目值],

name: [条目值],

age: [条目值],

sex: [条目值],

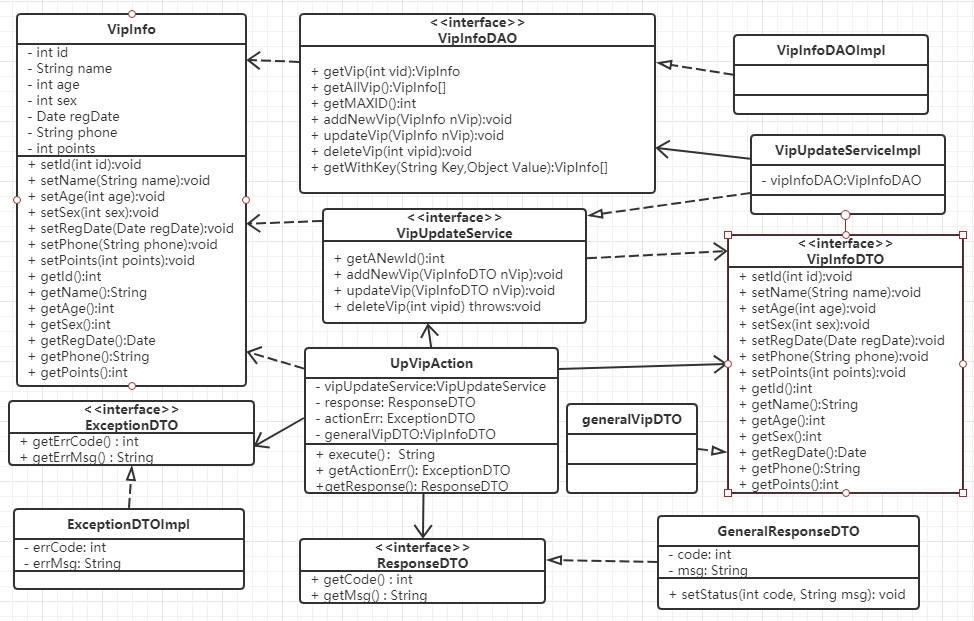
regDate: [条目值],

phone: [条目值],

points: [条目值]}

输出：JSON数据{code：[值],msg：[值]}；

#### 模块内部结构设计



#### 流程逻辑设计

### 删除会员信息

#### 程序描述

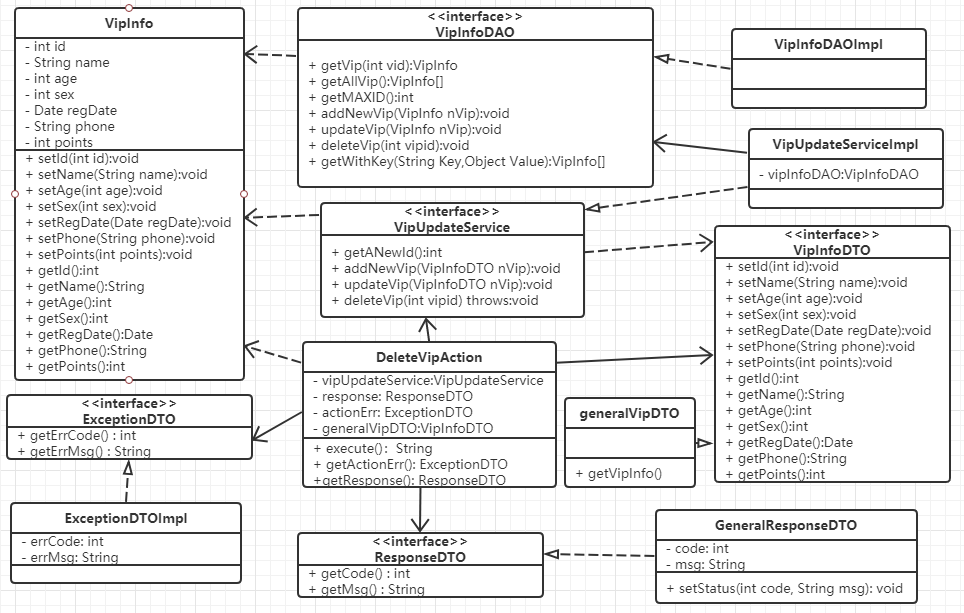
可以选择删除会员信息，在删除之前需要有是否确认删除的弹窗，避免失误操作。

#### 输入与输出

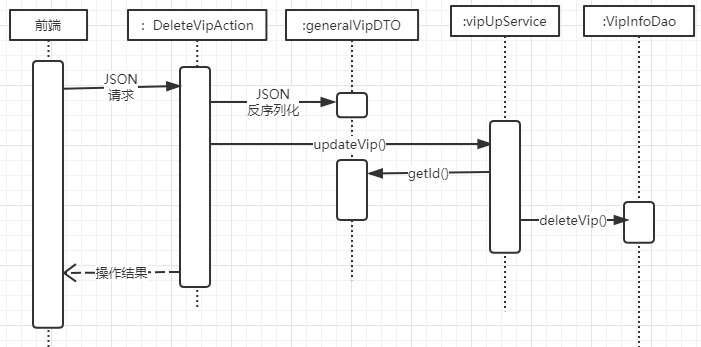
输入：JSON对象{id: [值]}

输出：JSON数据{code：[值],msg：[值]}；

#### 模块内部结构设计



#### 流程逻辑设计



### 注册新会员

#### 程序描述

注册新会员由超市员工操作，输入会员需要的所有信息，包括会员年龄，会员性别，会员姓名，会员手机号。然后自动生成会员ID和注册会员日期并初始化会员积分

#### 输入与输出

输入：JSON对象{id: [条目值],

name: [条目值],

age: [条目值],

sex: [条目值],

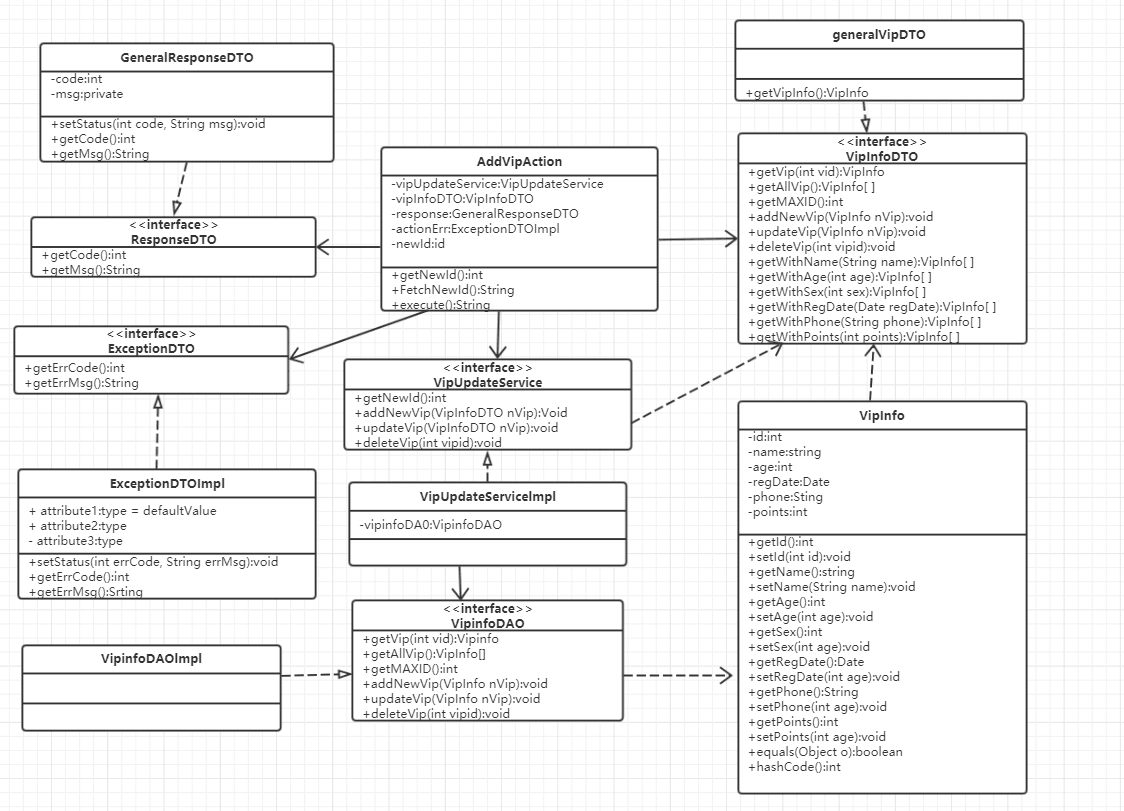
regDate: [条目值],

phone: [条目值],

points: [条目值]}

输出：JSON数据{code：[值],msg：[值]}；

#### 模块内部结构设计



#### C:\Users\lenovo\Documents\Tencent Files\361473023\Image\C2C\[{[{{F_GRX1IAIFS~VH6$}T.png流程逻辑设计

### 筛选消费记录

#### 程序描述

员工可以通过消费记录任意信息条目筛选消费记录

#### 输入与输出

JSON对象数据{key：[key值],[key值]：[条目值]}；

输出：JSON数组数据[{id: [条目值],

vipid: [条目值],

pointsChange: [条目值],

purDate: [条目值],

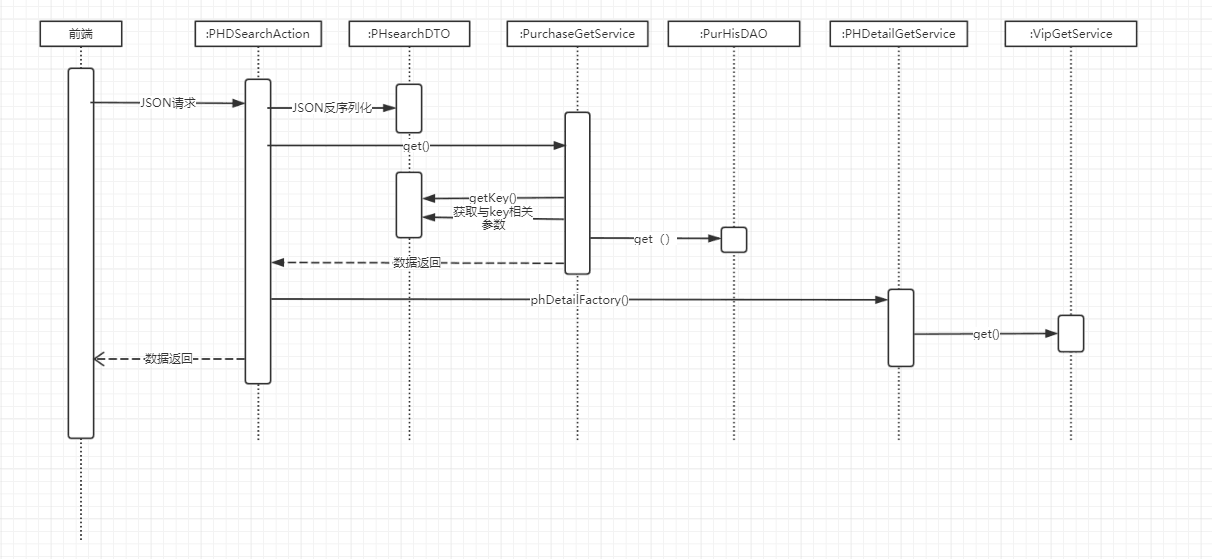
regDate: [条目值],

name: [条目值],

points: [条目值]},..]；

#### C:\Users\lenovo\Documents\Tencent Files\361473023\Image\C2C\5{}PJ_B3)YZCI%56T7T7L4U.png模块内部结构设计

#### 流程逻辑设计



## 逻辑表结构设计

* vipInfo表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Id(主键) | Name | Age | Sex | RegDate | Phone | Points |
| 整型 | 字符串型 | 整型 | 整型 | 日期类型 | 字符串型 | 整型 |

* purchaseHis表

|  |  |  |  |
| --- | --- | --- | --- |
| Id(主键) | Vipid（外键） | PointsChange | PurDate |
| 整型 | 整型 | 整型 | 日期类型 |

* userT表

|  |  |  |
| --- | --- | --- |
| Uid | Uname | Password |
| 整型 | 字符串型 | 字符串型 |

# 系统实现

## 配置说明

|  |  |
| --- | --- |
| 操作系统 | Windows10 |
| 开发工具 | IntelliJ IDEA 2018.3.5 (Ultimate Edition)  Visual Studio Code 1.32.3 |
| 版本管理工具 | Git 2.16.1 |
| JDK版本 | 1.8.0\_191 |
| 应用服务器 | Tomcat9.016 |
| 数据库 | SQL Server 2017 |

|  |  |
| --- | --- |
| Spring | 5.0.6.RELEASE |
| Hibernate | 5.3.1.Final |
| struts2 | 2.5.14.1 |
| Log4j | 2.11.1 |

|  |  |
| --- | --- |
| bulma | v0.7.4 |
| Font Awesome | Free 5.8.1 |
| Vue.js | v2.6.10 |
| axios | v0.18.0 |

## Spring配置-实现控制反转

Spring配置文件为 /项目目录/src/main/resources/applicationContext.xml。

该配置文件主要做以下几件事:

* 配置自动扫描组件的包，Spring将会根据注解将类注册为组件。
* 设置数据源连接(使用数据库驱动连接数据库)。
* 配置ORM框架与相应事务管理，注册可以操作model层模板类的工厂bean，本项目具体为hibernate sessionFactory的配置。
* 配置AOP切面编程

其中最核心的是控制反转，当通过配置文件与注解将类注册为组件后，可以通过Spring来做依赖注入：当一个对象需要使用另一个对象时，Spring会通过合适方式提供这个对象，使用者而不用自己去new。组件默认是单例的，也就是说一个组件被使用后并不会直接销毁而是留存在内存池中由spring管理，在另一个使用者对象需要时，该对象能被复用，减少创建对象以及垃圾回收的花销。

## model层实现-ORM映射

orm是将数据映射成对象的技术，本项目采用的是hibernate框架。

model层由一组POJO类模板组成，每个POJO类对应一个.hbm.xml配置文件。

Hibernate在读取数据库时，会根据model层类模板中的注解以及相应.hbm.xml文件中的配置，将读取的数据映射为POJO对象。

而hibernate的sessionFactory如6.2所述配置在applicationContext.xml中。

## dao层实现-封装sql查询

dao层每个表向上(Service层)提供一组接口，一般返回值为model包的POJO对象所组成的列表。内部实现则使用Hibernate提供的接口，具体流程为：

* 通过Spring依赖注入获取配置文件注册的sessionFactory组件。
* sessionFactory获取Session (Session会缓存一些之前的查询)。
* 通过session发起查询，查询结果为一个列表，列表容器内为按照model层配置的映射生成的POJO对象。



## Service层-核心业务逻辑

Service层的核心业务在于利用dao层提供的服务向上(Action)层提供三组接口：PurchaseGetService、VipGetService、VipUpdateService，负责基本的的增删改查。

除了基本的功能，Service层实现以下几个业务功能:

* 联合查询整合数据表:

由于 Hibernate的Session具有缓存查询结果的功能，所以将联合查询放在Service层，通过不同的DAO服务获取数据再拼接成前端需要的数据结构形式。

* 数据验证功能:

前端用户注册会员时是由系统分配id的，而前端传回的会员数据需与系统分配的id一致，VipUpdateService的实现VipUpdateServiceImpl会在action请求分配新id后维护该id值，并在action请求注册新会员信息时检查传入数据的id值。

* 通过切面编程做数据预处理:

Service层的VipGetAspect会拦截Service层所有获取会员信息的方法并查询相应消费记录，然后根据消费记录积分变动更新会员积分。

## Action层-通过Struts控制业务流程

Action层是引入struts2框架作用的控制器层,本项目Action包内Action类具有以下特征：

* 继承Struts2包的ActionSupport类，重写execute()方法；
* 实现Struts2包ModelDriven<T> 接口，实现getModel()方法；

继承ActionSupport是为了告知struts2业务逻辑所在的地方，struts2会在execute()执行之前与之后做若干准备与善后工作(面向切面)。

而实现ModelDriven接口是为了配合Struts2的json插件接受参数，该方法由ModelDriven拦截器调用，Struts2会将返回的对象压入值栈栈顶，而json插件在被拦截器调用时会取出栈顶对象并逐个该对象调用setter方法传入反序列化后的数据。

故一般Action在struts2中的拦截器配置如下:



每个Action的作用流程大同小异:

* Struts2根据前端请求根据配置分配Action；
* 若该Action配置了ModelDriven拦截器，调用该Action的getModel()方法，将返回的对象压入值栈栈顶；
* 若该Action配置了json拦截器，反序列化post请求传入的json数据，取出值栈栈顶的对象，逐个调用该对象的setter方法，传入数据；
* 调用execute()方法获取String类型返回值；
* 根据返回值匹配该Action配置的result；
* 若该result类型为json，根据配置的root参数，调用action相应字段的getter方法来序列化为json数据；
* 将json数据写入response.data，响应请求；

## DTO-数据承载模型

如6.6所描述，Action是通过ModelDriven接口去接收数据的，而ModelDriven需要返回的对象是如同model层一样的POJO，DTO包的对象就是负责以对象的形式接受前端发送过来的数据的，那么为何要单独分一组DTO而不是将其放在model包内？

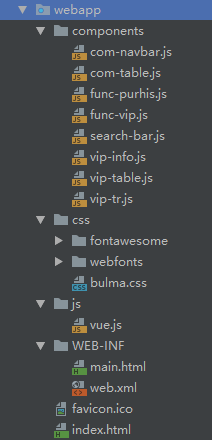
有以下几点理由：

* Model包的对象并未配置在Spring的引包路径中，model包由Hibernate负责生成，Action在不进行查询操作的情况下，需要model包的对象只能自己new。
* Model包的对象是在整个系统生命周期中多例的，而我们仅需要单例的数据承载对象 (可以避免多余的创建/垃圾回收花销)

单独分出的一组由Spring管理的只在Service层以上起作用、与DAO层以下的底层结构无关的、在整个应用生命周期里单例存在的POJO，就是DTO。

## 基于Vue.js的前端模块化实现

Vue.js为过于自由的JavaScript提供了一套有效的模块化模式，限于环境配置时间前端并没有采用webpack工具去做模块打包，但即使是采用Vue.component()函数去注册组件的形式，Vue的组件机制依然大幅度提高了JavaScript代码的可维护性。

本项目前端项目结构如图:

components目录下的js文件为一个Vue组件注册的函数调用，单个组件就是一个独立的MVVM模型。

得益于Vue 的双向数据绑定，结合css框架，使得每个组件可以通过ViewModel去控制View样式——页面样式能随数据变换而自适应，若不采用MVVM的模式，要实现同样的效果易伴随大量的事件监听与函数回调嵌套，将导致代码可读性、可维护性降低。

核心的业务功能体现在Main.html中，这是是一个SPA(单页面应用)，该页面中的JavaScript代码主要起到一个控制器的作用，负责与后端通信以及装配各种Vue组件。

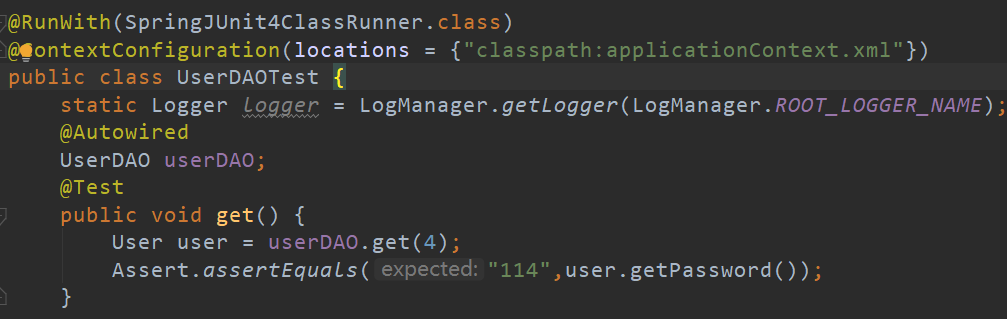
登录功能限制于开发时间并没有采用token机制，仅简单地利用跨域的特性去实现。Index.html为前端登录页面，其登录向后端Action发送请求，而Action验证输入后若通过便会返回跨域资源main.html(并不允许前端跨域请求而是直接返回跨域资源)

# 测试

测试主要以DAO层与Service层的单元测试为主

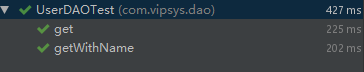
具体测试类源码在 项目目录\src\test\com\vipsys下的各包中

测试流程大致为：测试类持有一个接口引用，由Spring上下文自动注入相应实现，调用接口方法，通过Junit断言检查方法返回。

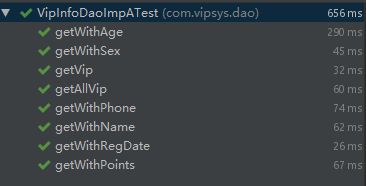


### DAO层接口测试

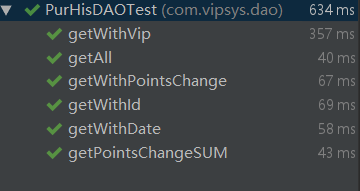
UserDAO测试:



VipInfoDAO测试：

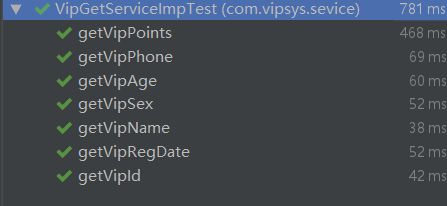


PurHisDAO接口测试:

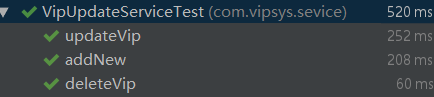


### Service层接口测试

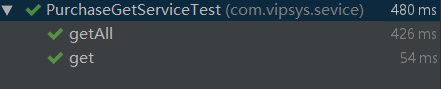
VipGetService测试:



vipUpdateService测试:



PurchaseGetService测试：



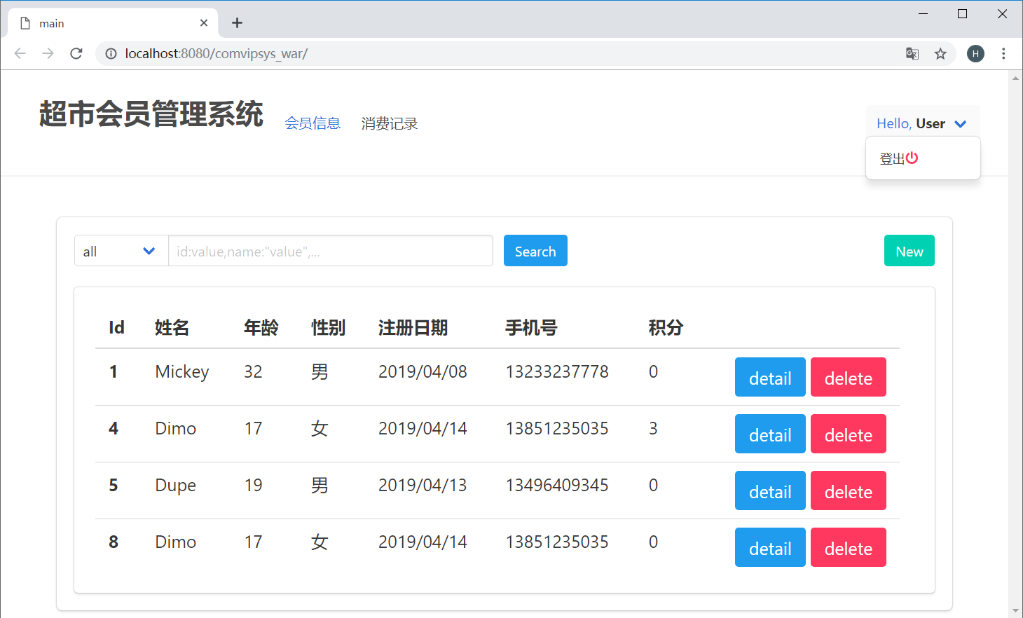
# 使用说明

## 登录

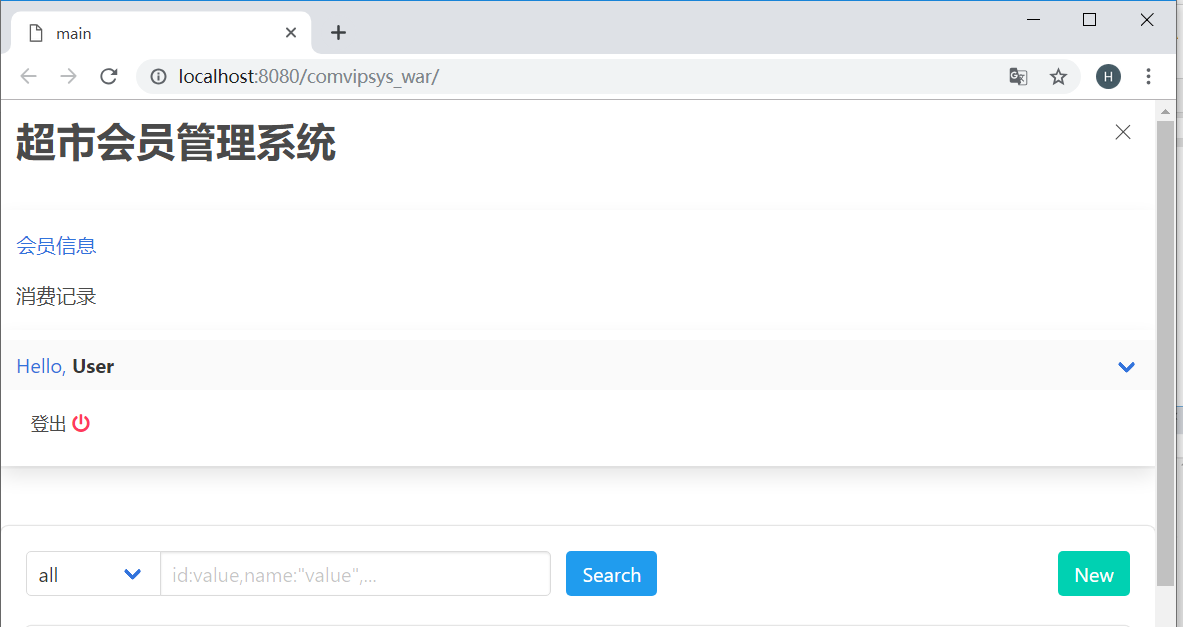
登录按照页面提示进行登录即可，登录成功后页面会自动刷新进入系统。

## 管理系统

管理系统大致界面如下，分为顶栏与主体部分功能模块。



顶栏会通过窗口大小收缩自适应。



### 会员信息管理

选中顶栏的“会员信息”会进入会员信息管理功能(该功能默认选中)。

该功能由搜索栏、新建按钮与表单数据组成。

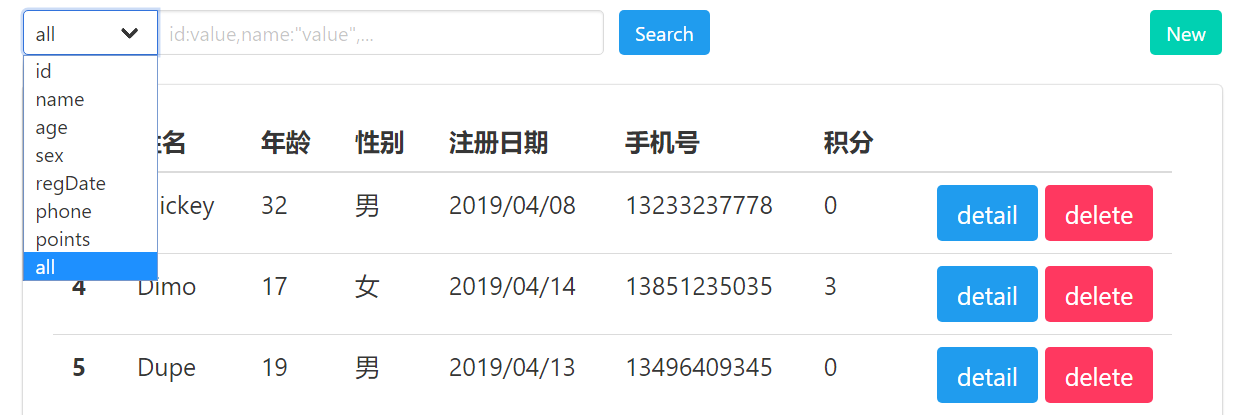
该功能启动时默认加载全部表单数据。

搜索栏提供了各表单字段的筛选选项，此外有个特别的all选项，选项为空时搜索所有条目，输入条件可实现聚合搜索，需要按输入框提示的格式输入。

点击Search按钮时会检查输入并发起搜索请求。

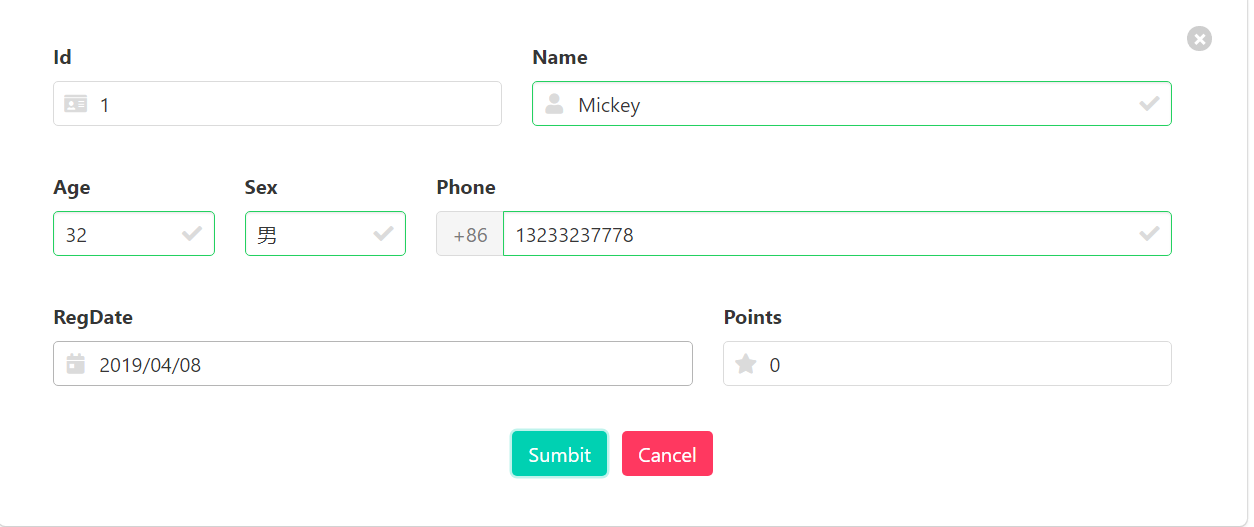
表单中的每个条目提供了detail与delete按钮以查看详情与删除服务。

功能图示如下:

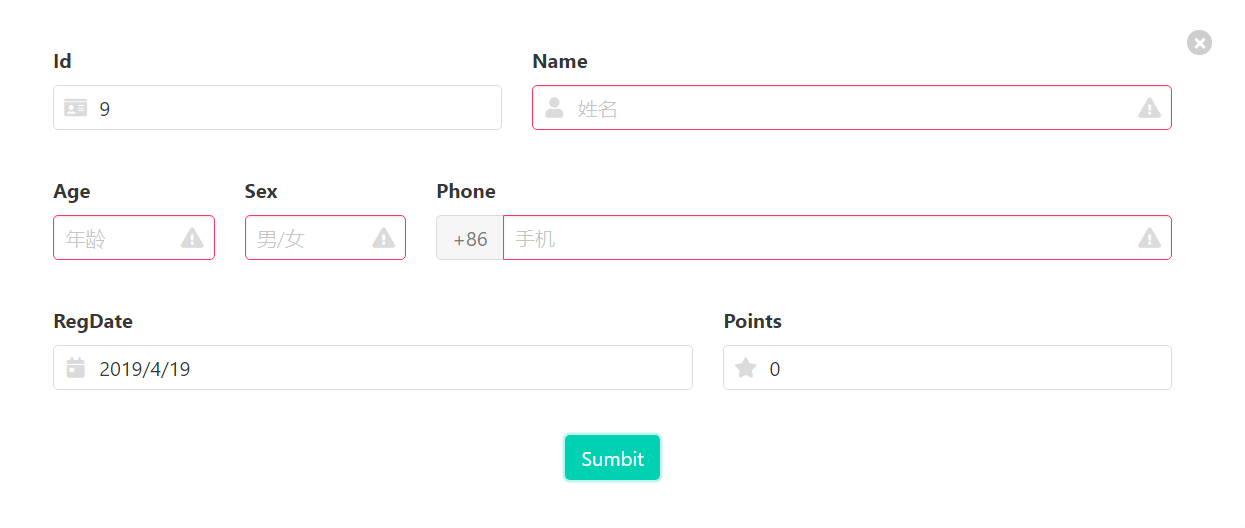


点击New或detail会进入会员信息页面

由Detail入口进入的界面显示已有信息详情况，在此界面可以点击edit按钮启用修改，输入修改点击sumbit进行提交时会检查输入，仅允许输入如图所示的检查输入的字段，其它字段由系统维护或在注册时生成。Cancel按钮会撤销所有未提交修改。



由New进入的界面与上述大同小异，其中id是由系统向后端查询生成的未占用id。



而上述界面右上角的x按钮为退回上级表格页面按钮，表格在查询后回被缓存，所以由x按钮退出后不会重新查询而是显示进入本页面前的状态。

### 消费记录查询

消费记录功能与8.2.1所述界面大致相同，不提供修改与新建功能，消费记录表由外部系统生成提供(超市会员消费进行消费)。

本界面显示的是对会员数据表与消费记录表进行联合查询后返回的数据表格。

本界面提供面向消费记录表的字段筛选功能。

