# 项目的系统功能

## 项目中需要有的功能 ——互联网的架构是非常复杂的

注册

登录

首页

搜索

商品详情页

购物车

订单（正向交易、逆向交易）

支付

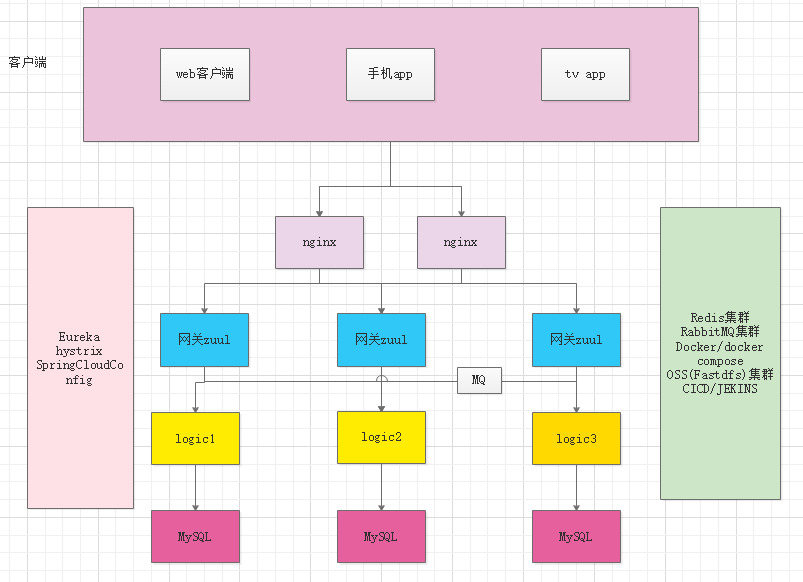
营销线：

秒杀（体会到高可用、高性能、高并发）

积分（京豆、淘金币）

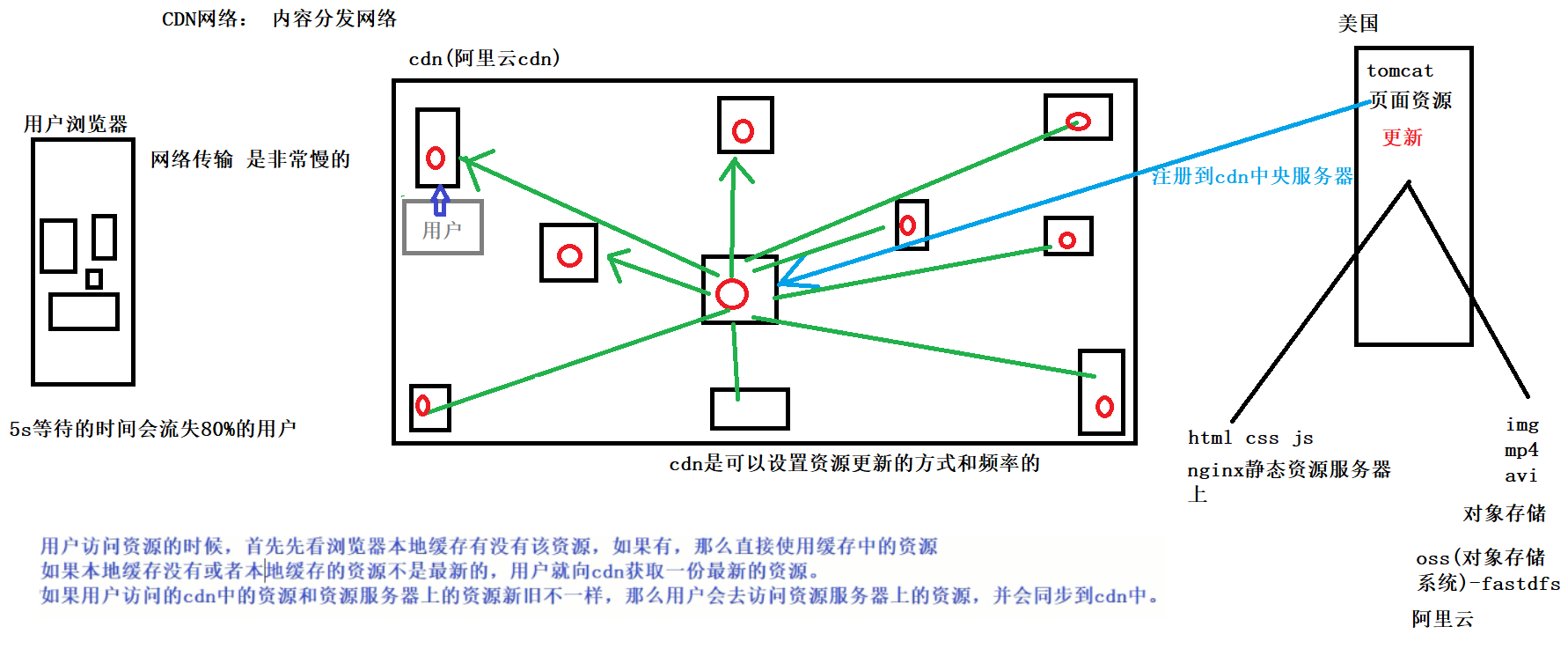
优惠券（拉新的一种手段）

## 架构图



## 前端页面的部署

### 1）使用内容分发网络



### 使用nginx对html css js做访问优化 ip和端口

（1）使用docker-compose搭建一台nginx虚拟主机

|  |
| --- |
| version: '3.1'  services:  nginx:  restart: always  image: nginx  container\_name: nginx  ports:  - 80:80  volumes:  - ./conf/nginx.conf:/etc/nginx/nginx.conf  - ./wwwroot:/usr/share/nginx/wwwroot |

1. 将静态资源存入到 wwwroot文件夹内
2. 在conf文件夹内创建nginx.conf配置文件

|  |
| --- |
| user nginx;  worker\_processes 1;  events {  worker\_connections 1024;  }  http {  include mime.types;  default\_type application/octet-stream;  sendfile on;  keepalive\_timeout 65;  server {  listen 80;  server\_name localhost;  # 解决nginx的跨域问题  add\_header Access-Control-Allow-Origin \*;  add\_header Access-Control-Allow-Headers X-Requested-With;  add\_header Access-Control-Allow-Methods GET,POST,OPTIONS;  location / {  root /usr/share/nginx/wwwroot/;  index index.html index.htm;  }  }  } |

### 使用OSS(FASTDFS)对象存储系统来存储图片、音频、视频

# 项目的具体实现——注册模块

## 注册模块的搭建

## Thymeleaf的介绍

目前用它来解决我们前端展示数据的问题，使用thymeleaf可以通过模板+数据的方式==得到页面，这个页面在有数据时展示数据，在没数据时，只展示模板。

## Thymeleaf的使用

1. 要在页面上的html标签中使用thymeleaf的标签库

|  |
| --- |
| <**html xmlns:th="http://www.thymeleaf.org"**> |

1. 使用Thymeleaf表达式获得后端的数据

一定要注意： thymeleaf是必须写在标签内的

1. username:zhangsan ${username}

|  |
| --- |
| <**a href="# "**>联系<**span th:text="${username}"**>我们</**span**></**a**> **这样可以** <**a href="# "**>联系${username}</**a**> **这样不行** |

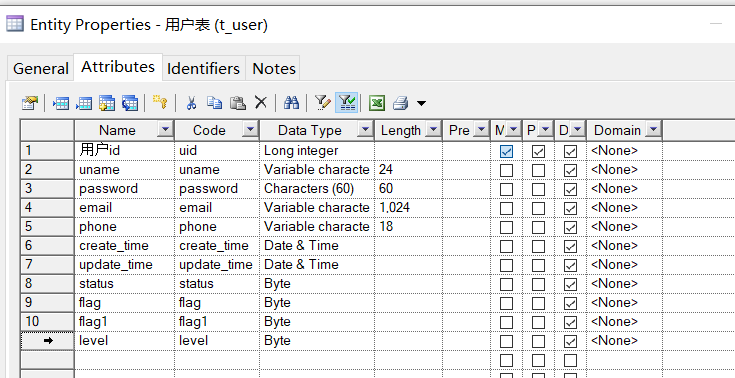
1. 获得对象中的属性
2. 如何执行if判断
3. 如何执行循环遍历

|  |
| --- |
| @RequestMapping(**"/demo"**) **public** String demo(Model model){  Student stu1 = **new** Student(**"zhangsan"**,20);  Student stu2 = **new** Student(**"lisi"**,22);   List<Student> stus = **new** ArrayList<>();  stus.add(stu1);  stus.add(stu2);   model.addAttribute(**"stus"**,stus);   **return "demo"**; } |

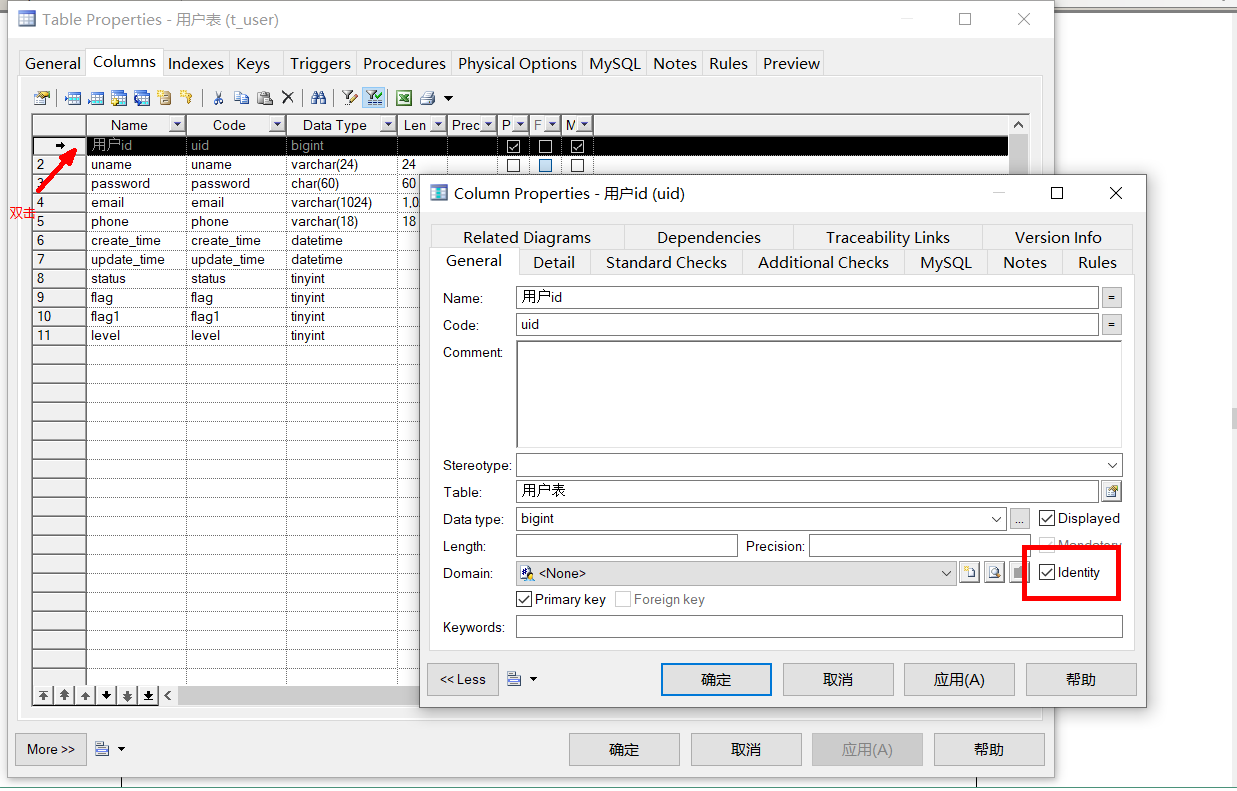
|  |
| --- |
| <**table border="1px" width="100%"**>   <**tr**>  <**td**>姓名</**td**>  <**td**>年龄</**td**>  </**tr**>  <**tr th:each="stu : ${stus}" th:if="${stu.age>20}"**> if表示只有age大于20才会显示  <**td th:text="${stu.name}"**>姓名</**td**>  <**td th:text="${stu.age}"**>年龄</**td**>  </**tr**>  </**table**> |

# 数据库的设计

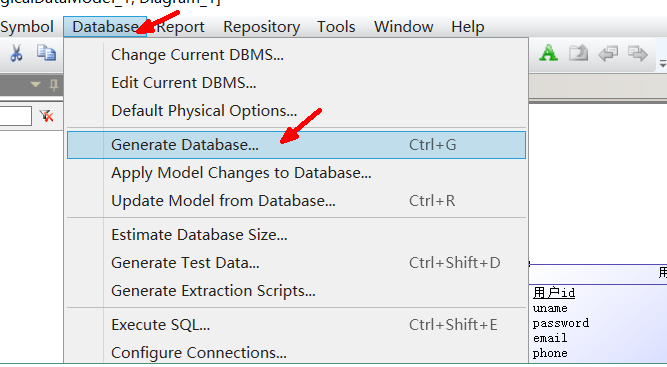
## 1.设计用户表的逻辑模型



## 转换成物理模型后设置主键自增

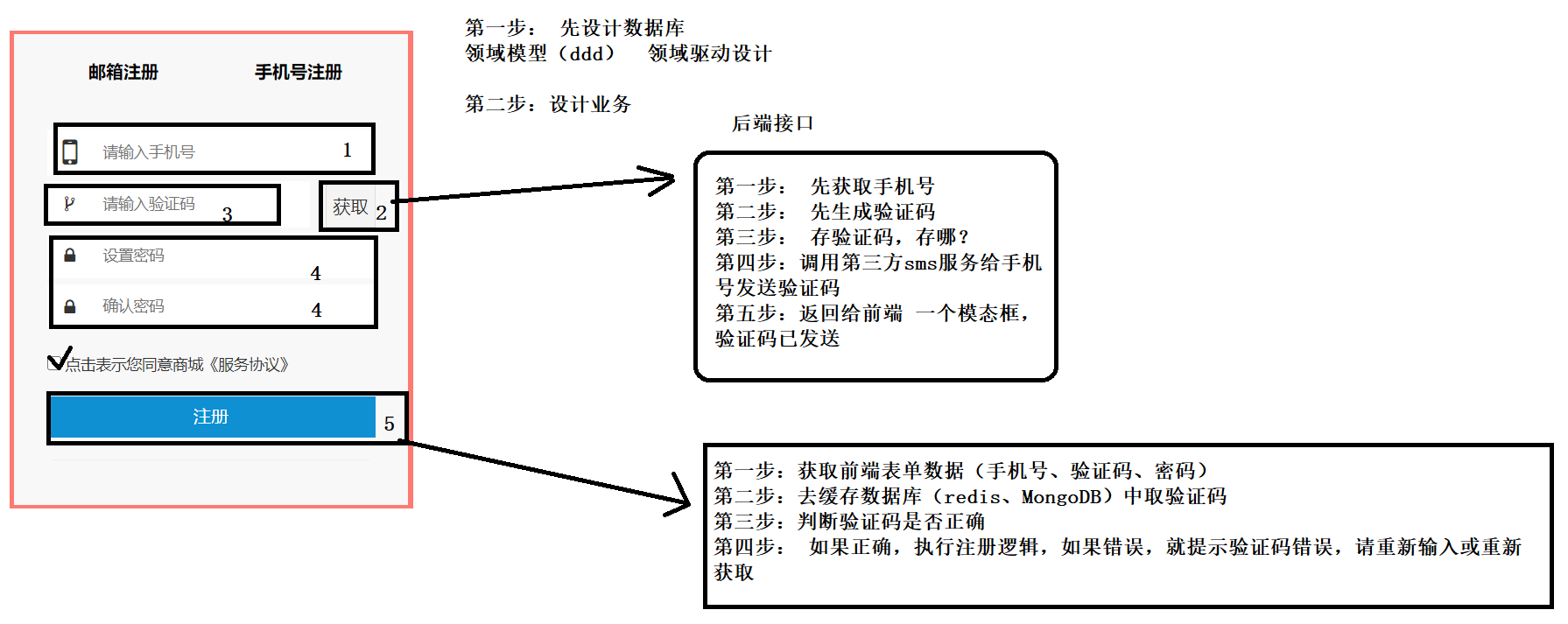


生成好sql语句导入到数据库中



# 手机号注册

## 流程梳理



## 代码实现

|  |
| --- |
| **package** com.qf.my.shop.regist.service.impl;  **import** com.qf.common.constant.RedisConstant; **import** com.qf.common.dto.ResultBean; **import** com.qf.common.util.RestHttpUtil; **import** com.qf.my.shop.regist.service.IRegistService; **import** com.qf.my.shop.regist.util.StringUtil; **import** org.springframework.beans.factory.annotation.Autowired; **import** org.springframework.http.HttpHeaders; **import** org.springframework.stereotype.Service; **import** org.springframework.util.LinkedMultiValueMap; **import** org.springframework.util.MultiValueMap; **import** org.springframework.web.client.RestTemplate;  @Service **public class** RegistServiceImpl **implements** IRegistService {   @Autowired  **private** RestTemplate **restTemplate**;   @Override  **public** ResultBean getSmsCode(String phone) {  String smsCode = StringUtil.*getSmsCode*();  *//=========把验证码存到redis中==================  //regist:手机号===验证码  //组织redis中的键* String pre = RedisConstant.***REGIST\_PRE***;  String redisKey = StringUtil.*getRedisKey*(pre, phone);   *//封装请求头* HttpHeaders headers = **new** HttpHeaders();  MultiValueMap<String, Object> map = **new** LinkedMultiValueMap<>();  map.add(**"key"**, redisKey);  map.add(**"value"**, smsCode);   String url = **"http://CACHE-DATA-ACCESS/cache/data/insert"**;   RestHttpUtil.*requestByPost*(headers, map, url, **restTemplate**);    *//=========使用短信服务来发送短信==================* HttpHeaders headers1 = **new** HttpHeaders();  MultiValueMap<String, Object> map1 = **new** LinkedMultiValueMap<>();  map1.add(**"phone"**, phone);  map1.add(**"code"**, smsCode);  String url1 = **"http://SMS-SERVER/sms/send"**;  ResultBean resultBean = RestHttpUtil.*requestByPost*(headers1, map1, url1, **restTemplate**);  **return** resultBean;  } } |

1. 画流程图
2. 画时序图
3. 画用例图

# RestTemplate的使用

## getForObject

|  |
| --- |
| String username = restTemplate.getForObject("http://localhost:8080/getNameById?id={1}&name={2}",String.class,new Object[]{id,name});  Map<String,Object> params = new HashMap<>();  params.put("id",id);  params.put("name",name);  User user = restTemplate.getForObject("http://localhost:8080/getUserById?id={id}&name={name}",User.class,params);  UriComponents uriComponents = UriComponentsBuilder.fromUriString("http://localhost:8080/getUserByname?name={name}").build().expand(name).encode();  URI uri = uriComponents.toUri();  String response = restTemplate.getForObject(uri,String.class); |

## postForObject

|  |
| --- |
| HttpHeaders headers = new HttpHeaders();  MultiValueMap<String, Object> parammap = new LinkedMultiValueMap<>();  parammap.add("privilege",privilege);    HttpEntity<Map> entity = new HttpEntity<>(parammap,headers);    User user = restTemplate.postForObject(userService+"/queryUserByPrivilege",entity,User.class);  List<String> menulist = restTemplate.postForObject("http://localhost:8080/queryLoginPrivilegeByUser",user,List.class); |

# 阿里云的短信服务

1. 登录阿里云短信服务控制台
2. 登录用户中心进行充值



1. 添加签名

在国内消息中添加签名



测试用的话，签名就是ABC商城

1. 添加模板

