1 学习目标

完成用户注册功能

完成redis数据库搭建

完成短信发送功能

熟练使用redis

2 项目搭建

2.1 service-api

2.1.1 新建mingrui-shop-service-api-user

2.1.2 新建包com.baidu.shop.entity

2.1.3 包下新建UserEntity

```java
import lombok.Data;

import javax.persistence.Id;
import javax.persistence.Table;
import java.util.Date;

/**
 * @ClassName UserEntity
 * @Description: TODO
 * @Author shenyaqi
 * @Date 2020/9/20
 * @Version V1.0
 **/
@Table(name = "tb_user")
@Data
public class UserEntity {

    @Id
    private Integer id;

    private String username;

    private String password;

    private String phone;

    private Date created;

    private String salt;
}
```

2.1.4 新建包com.baidu.shop.dto

2.1.5 包下新建UserDTO

```java
import com.baidu.shop.validate.group.MingruiOperation;
import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;
import lombok.Data;

import javax.validation.constraints.NotNull;
import java.util.Date;

/**
 * @ClassName UserDTO
 * @Description: TODO
 * @Author shenyaqi
 * @Date 2020/9/20
 * @Version V1.0
 **/
@Data
@ApiModel(value = "用户DTO")
public class UserDTO {

    @ApiModelProperty(value = "用户主键",example = "1")
    @NotNull(message = "主键不能为空", groups = {MingruiOperation.Update.class})
    private Integer id;

    @ApiModelProperty(value = "账户")
    @NotNull(message = "账户不能为空", groups = {MingruiOperation.Add.class})
    private String username;

    @ApiModelProperty(value = "密码")
    @NotNull(message = "密码不能为空", groups = {MingruiOperation.Add.class})
    private String password;

    @ApiModelProperty(value = "手机号")
    @NotNull(message = "手机号不能为空", groups = {MingruiOperation.Add.class})
    private String phone;

    private Date created;

    private String salt;
}
```

2.1.6 新建包com.baidu.shop.config

2.1.7 新建MrSwagger2Config

```java
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import springfox.documentation.builders.ApiInfoBuilder;
import springfox.documentation.builders.PathSelectors;
import springfox.documentation.builders.RequestHandlerSelectors;
import springfox.documentation.service.ApiInfo;
import springfox.documentation.service.Contact;
import springfox.documentation.spi.DocumentationType;
import springfox.documentation.spring.web.plugins.Docket;
import springfox.documentation.swagger2.annotations.EnableSwagger2;

/**
 * @ClassName MrSwagger2Config
 * @Description: TODO
 * @Author shenyaqi
 * @Date 2020/8/17
 * @Version V1.0
 **/
@Configuration
@EnableSwagger2
public class MrSwagger2Config {

    @Bean
    public Docket createRestApi(){
        return new Docket(DocumentationType.SWAGGER_2)
                .apiInfo(this.apiInfo())
                .select()
                .apis(RequestHandlerSelectors.basePackage("com.baidu"))
                .paths(PathSelectors.any())
                .build();
    }

    private ApiInfo apiInfo(){
        return new ApiInfoBuilder()
                //标题
                .title("明瑞SWAGGER2标题")
                //条款地址
                .termsOfServiceUrl("http://www.baidu.com")
                //联系方式-->有String参数的方法但是已经过时，所以不推荐使用
                .contact(new
Contact("shenyaqi","baidu.com","shenyaqiii@163.com"))
                //版本
                .version("v1.0")
                //项目描述
                .description("描述")
                //创建API基本信息
                .build();
    }
}
```

2.1.8 新建包com.baidu.shop.service

2.1.9 新建UserService

```
import com.alibaba.fastjson.JSONObject;
import com.baidu.shop.base.Result;
import com.baidu.shop.dto.UserDTO;
import io.swagger.annotations.Api;
import io.swagger.annotations.ApiOperation;
import org.springframework.web.bind.annotation.PostMapping;

/**
```

```
 * @ClassName userService
 * @Description: TODO
 * @Author shenyaqi
 * @Date 2020/9/20
 * @Version V1.0
 **/
@Api(tags = "用户接口")
public interface UserService {

    @ApiOperation(value = "用户注册")
    @PostMapping(value = "user/register")
    Result<JSONObject> register(UserDTO userDTO);
}
```

2.2 common-core

2.2.1 utils包下新建BCryptUtil

```
import java.io.UnsupportedEncodingException;
import java.security.SecureRandom;

/**
 * @ClassName BCryptUtil
 * @Description: TODO
 * @Author shenyaqi
 * @Date 2020/9/20
 * @Version V1.0
 **/
public class BCryptUtil {

    // BCrypt parameters
    private static final int GENSALT_DEFAULT_LOG2_ROUNDS = 10;
    private static final int BCRYPT_SALT_LEN = 16;

    // Blowfish parameters
    private static final int BLOWFISH_NUM_ROUNDS = 16;

    // Initial contents of key schedule
    private static final int P_orig[] = {
            0x243f6a88, 0x85a308d3, 0x13198a2e, 0x03707344,
            0xa4093822, 0x299f31d0, 0x082efa98, 0xec4e6c89,
            0x452821e6, 0x38d01377, 0xbe5466cf, 0x34e90c6c,
            0xc0ac29b7, 0xc97c50dd, 0x3f84d5b5, 0xb5470917,
            0x9216d5d9, 0x8979fb1b
    };
    private static final int S_orig[] = {
            0xd1310ba6, 0x98dfb5ac, 0x2ffd72db, 0xd01adfb7,
            0xb8e1afed, 0x6a267e96, 0xba7c9045, 0xf12c7f99,
            0x24a19947, 0xb3916cf7, 0x0801f2e2, 0x858efc16,
            0x636920d8, 0x71574e69, 0xa458fea3, 0xf4933d7e,
            0x0d95748f, 0x728eb658, 0x718bcd58, 0x82154aee,
            0x7b54a41d, 0xc25a59b5, 0x9c30d539, 0x2af26013,
            0xc5d1b023, 0x286085f0, 0xca417918, 0xb8db38ef,
            0x8e79dcb0, 0x603a180e, 0x6c9e0e8b, 0xb01e8a3e,
            0xd71577c1, 0xbd314b27, 0x78af2fda, 0x55605c60,
            0xe65525f3, 0xaa55ab94, 0x57489862, 0x63e81440,
            0x55ca396a, 0x2aab10b6, 0xb4cc5c34, 0x1141e8ce,
            0xa15486af, 0x7c72e993, 0xb3ee1411, 0x636fbc2a,
            0x2ba9c55d, 0x741831f6, 0xce5c3e16, 0x9b87931e,
            0xafd6ba33, 0x6c24cf5c, 0x7a325381, 0x28958677,
            0x3b8f4898, 0x6b4bb9af, 0xc4bfe81b, 0x66282193,
            0x61d809cc, 0xfb21a991, 0x487cac60, 0x5dec8032,
            0xef845d5d, 0xe98575b1, 0xdc262302, 0xeb651b88,
            0x23893e81, 0xd396acc5, 0x0f6d6ff3, 0x83f44239,
            0x2e0b4482, 0xa4842004, 0x69c8f04a, 0x9e1f9b5e,
            0x21c66842, 0xf6e96c9a, 0x670c9c61, 0xabd388f0,
            0x6a51a0d2, 0xd8542f68, 0x960fa728, 0xab5133a3,
            0x6eef0b6c, 0x137a3be4, 0xba3bf050, 0x7efb2a98,
            0xa1f1651d, 0x39af0176, 0x66ca593e, 0x82430e88,
            0x8cee8619, 0x456f9fb4, 0x7d84a5c3, 0x3b8b5ebe,
            0xe06f75d8, 0x85c12073, 0x401a449f, 0x56c16aa6,
            0x4ed3aa62, 0x363f7706, 0x1bfedf72, 0x429b023d,
            0x37d0d724, 0xd00a1248, 0xdb0fead3, 0x49f1c09b,
            0x075372c9, 0x80991b7b, 0x25d479d8, 0xf6e8def7,
            0xe3fe501a, 0xb6794c3b, 0x976ce0bd, 0x04c006ba,
            0xc1a94fb6, 0x409f60c4, 0x5e5c9ec2, 0x196a2463,
            0x68fb6faf, 0x3e6c53b5, 0x1339b2eb, 0x3b52ec6f,
            0x6dfc511f, 0x9b30952c, 0xcc814544, 0xaf5ebd09,
            0xbee3d004, 0xde334afd, 0x660f2807, 0x192e4bb3,
            0xc0cba857, 0x45c8740f, 0xd20b5f39, 0xb9d3fbdb,
            0x5579c0bd, 0x1a60320a, 0xd6a100c6, 0x402c7279,
            0x679f25fe, 0xfb1fa3cc, 0x8ea5e9f8, 0xdb3222f8,
            0x3c7516df, 0xfd616b15, 0x2f501ec8, 0xad0552ab,
            0x323db5fa, 0xfd238760, 0x53317b48, 0x3e00df82,
            0x9e5c57bb, 0xca6f8ca0, 0x1a87562e, 0xdf1769db,
            0xd542a8f6, 0x287effc3, 0xac6732c6, 0x8c4f5573,
            0x695b27b0, 0xbbca58c8, 0xe1ffa35d, 0xb8f011a0,
            0x10fa3d98, 0xfd2183b8, 0x4afcb56c, 0x2dd1d35b,
            0x9a53e479, 0xb6f84565, 0xd28e49bc, 0x4bfb9790,
            0xe1ddf2da, 0xa4cb7e33, 0x62fb1341, 0xcee4c6e8,
            0xef20cada, 0x36774c01, 0xd07e9efe, 0x2bf11fb4,
            0x95dbda4d, 0xae909198, 0xeaad8e71, 0x6b93d5a0,
            0xd08ed1d0, 0xafc725e0, 0x8e3c5b2f, 0x8e7594b7,
            0x8ff6e2fb, 0xf2122b64, 0x8888b812, 0x900df01c,
            0x4fad5ea0, 0x688fc31c, 0xd1cff191, 0xb3a8c1ad,
            0x2f2f2218, 0xbe0e1777, 0xea752dfe, 0x8b021fa1,
            0xe5a0cc0f, 0xb56f74e8, 0x18acf3d6, 0xce89e299,
            0xb4a84fe0, 0xfd13e0b7, 0x7cc43b81, 0xd2ada8d9,
            0x165fa266, 0x80957705, 0x93cc7314, 0x211a1477,
            0xe6ad2065, 0x77b5fa86, 0xc75442f5, 0xfb9d35cf,
            0xebcdaf0c, 0x7b3e89a0, 0xd6411bd3, 0xae1e7e49,
            0x00250e2d, 0x2071b35e, 0x226800bb, 0x57b8e0af,
            0x2464369b, 0xf009b91e, 0x5563911d, 0x59dfa6aa,
            0x78c14389, 0xd95a537f, 0x207d5ba2, 0x02e5b9c5,
            0x83260376, 0x6295cfa9, 0x11c81968, 0x4e734a41,
            0xb3472dca, 0x7b14a94a, 0x1b510052, 0x9a532915,
            0xd60f573f, 0xbc9bc6e4, 0x2b60a476, 0x81e67400,
            0x08ba6fb5, 0x571be91f, 0xf296ec6b, 0x2a0dd915,
            0xb6636521, 0xe7b9f9b6, 0xff34052e, 0xc5855664,
            0x53b02d5d, 0xa99f8fa1, 0x08ba4799, 0x6e85076a,
            0x4b7a70e9, 0xb5b32944, 0xdb75092e, 0xc4192623,
            0xad6ea6b0, 0x49a7df7d, 0x9cee60b8, 0x8fedb266,
            0xecaa8c71, 0x699a17ff, 0x5664526c, 0xc2b19ee1,
            0x193602a5, 0x75094c29, 0xa0591340, 0xe4183a3e,
            0x3f54989a, 0x5b429d65, 0x6b8fe4d6, 0x99f73fd6,
            0xa1d29c07, 0xefe830f5, 0x4d2d38e6, 0xf0255dc1,
            0x4cdd2086, 0x8470eb26, 0x6382e9c6, 0x021ecc5e,
            0x09686b3f, 0x3ebaefc9, 0x3c971814, 0x6b6a70a1,
            0x687f3584, 0x52a0e286, 0xb79c5305, 0xaa500737,
            0x3e07841c, 0x7fdeae5c, 0x8e7d44ec, 0x5716f2b8,
            0xb03ada37, 0xf0500c0d, 0xf01c1f04, 0x0200b3ff,
            0xae0cf51a, 0x3cb574b2, 0x25837a58, 0xdc0921bd,
            0xd19113f9, 0x7ca92ff6, 0x94324773, 0x22f54701,
            0x3ae5e581, 0x37c2dadc, 0xc8b57634, 0x9af3dda7,
            0xa9446146, 0x0fd0030e, 0xecc8c73e, 0xa4751e41,
            0xe238cd99, 0x3bea0e2f, 0x3280bba1, 0x183eb331,
            0x4e548b38, 0x4f6db908, 0x6f420d03, 0xf60a04bf,
            0x2cb81290, 0x24977c79, 0x5679b072, 0xbcaf89af,
            0xde9a771f, 0xd9930810, 0xb38bae12, 0xdccf3f2e,
            0x5512721f, 0x2e6b7124, 0x501adde6, 0x9f84cd87,
            0x7a584718, 0x7408da17, 0xbc9f9abc, 0xe94b7d8c,
            0xec7aec3a, 0xdb851dfa, 0x63094366, 0xc464c3d2,
            0xef1c1847, 0x3215d908, 0xdd433b37, 0x24c2ba16,
            0x12a14d43, 0x2a65c451, 0x50940002, 0x133ae4dd,
            0x71dff89e, 0x10314e55, 0x81ac77d6, 0x5f11199b,
            0x043556f1, 0xd7a3c76b, 0x3c11183b, 0x5924a509,
            0xf28fe6ed, 0x97f1fbfa, 0x9ebabf2c, 0x1e153c6e,
            0x86e34570, 0xeae96fb1, 0x860e5e0a, 0x5a3e2ab3,
            0x771fe71c, 0x4e3d06fa, 0x2965dcb9, 0x99e71d0f,
            0x803e89d6, 0x5266c825, 0x2e4cc978, 0x9c10b36a,
            0xc6150eba, 0x94e2ea78, 0xa5fc3c53, 0x1e0a2df4,
            0xf2f74ea7, 0x361d2b3d, 0x1939260f, 0x19c27960,
            0x5223a708, 0xf71312b6, 0xebadfe6e, 0xeac31f66,
            0xe3bc4595, 0xa67bc883, 0xb17f37d1, 0x018cff28,
            0xc332ddef, 0xbe6c5aa5, 0x65582185, 0x68ab9802,
            0xeecea50f, 0xdb2f953b, 0x2aef7dad, 0x5b6e2f84,
            0x1521b628, 0x29076170, 0xecdd4775, 0x619f1510,
            0x13cca830, 0xeb61bd96, 0x0334fe1e, 0xaa0363cf,
            0xb5735c90, 0x4c70a239, 0xd59e9e0b, 0xcbaade14,
            0xeecc86bc, 0x60622ca7, 0x9cab5cab, 0xb2f3846e,
            0x648b1eaf, 0x19bdf0ca, 0xa02369b9, 0x655abb50,
            0x40685a32, 0x3c2ab4b3, 0x319ee9d5, 0xc021b8f7,
            0x9b540b19, 0x875fa099, 0x95f7997e, 0x623d7da8,
            0xf837889a, 0x97e32d77, 0x11ed935f, 0x16681281,
            0x0e358829, 0xc7e61fd6, 0x96dedfa1, 0x7858ba99,
            0x57f584a5, 0x1b227263, 0x9b83c3ff, 0x1ac24696,
            0xcdb30aeb, 0x532e3054, 0x8fd948e4, 0x6dbc3128,
            0x58ebf2ef, 0x34c6ffea, 0xfe28ed61, 0xee7c3c73,
            0x5d4a14d9, 0xe864b7e3, 0x42105d14, 0x203e13e0,
            0x45eee2b6, 0xa3aaabea, 0xdb6c4f15, 0xfacb4fd0,
            0xc742f442, 0xef6abbb5, 0x654f3b1d, 0x41cd2105,
            0xd81e799e, 0x86854dc7, 0xe44b476a, 0x3d816250,
            0xcf62a1f2, 0x5b8d2646, 0xfc8883a0, 0xc1c7b6a3,
            0x7f1524c3, 0x69cb7492, 0x47848a0b, 0x5692b285,
            0x095bbf00, 0xad19489d, 0x1462b174, 0x23820e00,
            0x58428d2a, 0x0c55f5ea, 0x1dadf43e, 0x233f7061,
            0x3372f092, 0x8d937e41, 0xd65fecf1, 0x6c223bdb,
            0x7cde3759, 0xcbee7460, 0x4085f2a7, 0xce77326e,
            0xa6078084, 0x19f8509e, 0xe8efd855, 0x61d99735,
            0xa969a7aa, 0xc50c06c2, 0x5a04abfc, 0x800bcadc,
            0x9e447a2e, 0xc3453484, 0xfdd56705, 0x0e1e9ec9,
            0xdb73dbd3, 0x105588cd, 0x675fda79, 0xe3674340,
            0xc5c43465, 0x713e38d8, 0x3d28f89e, 0xf16dff20,
            0x153e21e7, 0x8fb03d4a, 0xe6e39f2b, 0xdb83adf7,
            0xe93d5a68, 0x948140f7, 0xf64c261c, 0x94692934,
            0x411520f7, 0x7602d4f7, 0xbcf46b2e, 0xd4a20068,
            0xd4082471, 0x3320f46a, 0x43b7d4b7, 0x500061af,
            0x1e39f62e, 0x97244546, 0x14214f74, 0xbf8b8840,
            0x4d95fc1d, 0x96b591af, 0x70f4ddd3, 0x66a02f45,
            0xbfbc09ec, 0x03bd9785, 0x7fac6dd0, 0x31cb8504,
            0x96eb27b3, 0x55fd3941, 0xda2547e6, 0xabca0a9a,
            0x28507825, 0x530429f4, 0x0a2c86da, 0xe9b66dfb,
            0x68dc1462, 0xd7486900, 0x680ec0a4, 0x27a18dee,
            0x4f3ffea2, 0xe887ad8c, 0xb58ce006, 0x7af4d6b6,
            0xaace1e7c, 0xd3375fec, 0xce78a399, 0x406b2a42,
            0x20fe9e35, 0xd9f385b9, 0xee39d7ab, 0x3b124e8b,
            0x1dc9faf7, 0x4b6d1856, 0x26a36631, 0xeae397b2,
            0x3a6efa74, 0xdd5b4332, 0x6841e7f7, 0xca7820fb,
            0xfb0af54e, 0xd8feb397, 0x454056ac, 0xba489527,
            0x55533a3a, 0x20838d87, 0xfe6ba9b7, 0xd096954b,
            0x55a867bc, 0xa1159a58, 0xcca92963, 0x99e1db33,
            0xa62a4a56, 0x3f3125f9, 0x5ef47e1c, 0x9029317c,
            0xfdf8e802, 0x04272f70, 0x80bb155c, 0x05282ce3,
            0x95c11548, 0xe4c66d22, 0x48c1133f, 0xc70f86dc,
            0x07f9c9ee, 0x41041f0f, 0x404779a4, 0x5d886e17,
            0x325f51eb, 0xd59bc0d1, 0xf2bcc18f, 0x41113564,
            0x257b7834, 0x602a9c60, 0xdff8e8a3, 0x1f636c1b,
            0x0e12b4c2, 0x02e1329e, 0xaf664fd1, 0xcad18115,
            0x6b2395e0, 0x333e92e1, 0x3b240b62, 0xeebeb922,
            0x85b2a20e, 0xe6ba0d99, 0xde720c8c, 0x2da2f728,
            0xd0127845, 0x95b794fd, 0x647d0862, 0xe7ccf5f0,
            0x5449a36f, 0x877d48fa, 0xc39dfd27, 0xf33e8d1e,
            0x0a476341, 0x992eff74, 0x3a6f6eab, 0xf4f8fd37,
            0xa812dc60, 0xa1ebddf8, 0x991be14c, 0xdb6e6b0d,
            0xc67b5510, 0x6d672c37, 0x2765d43b, 0xdcd0e804,
            0xf1290dc7, 0xcc00ffa3, 0xb5390f92, 0x690fed0b,
            0x667b9ffb, 0xcedb7d9c, 0xa091cf0b, 0xd9155ea3,
            0xbb132f88, 0x515bad24, 0x7b9479bf, 0x763bd6eb,
            0x37392eb3, 0xcc115979, 0x8026e297, 0xf42e312d,
            0x6842ada7, 0xc66a2b3b, 0x12754ccc, 0x782ef11c,
            0x6a124237, 0xb79251e7, 0x06a1bbe6, 0x4bfb6350,
            0x1a6b1018, 0x11caedfa, 0x3d25bdd8, 0xe2e1c3c9,
            0x44421659, 0x0a121386, 0xd90cec6e, 0xd5abea2a,
            0x64af674e, 0xda86a85f, 0xbebfe988, 0x64e4c3fe,
            0x9dbc8057, 0xf0f7c086, 0x60787bf8, 0x6003604d,
            0xd1fd8346, 0xf6381fb0, 0x7745ae04, 0xd736fccc,
            0x83426b33, 0xf01eab71, 0xb0804187, 0x3c005e5f,
            0x77a057be, 0xbde8ae24, 0x55464299, 0xbf582e61,
            0x4e58f48f, 0xf2ddfda2, 0xf474ef38, 0x8789bdc2,
            0x5366f9c3, 0xc8b38e74, 0xb475f255, 0x46fcd9b9,
            0x7aeb2661, 0x8b1ddf84, 0x846a0e79, 0x915f95e2,
            0x466e598e, 0x20b45770, 0x8cd55591, 0xc902de4c,
            0xb90bace1, 0xbb8205d0, 0x11a86248, 0x7574a99e,
            0xb77f19b6, 0xe0a9dc09, 0x662d09a1, 0xc4324633,
            0xe85a1f02, 0x09f0be8c, 0x4a99a025, 0x1d6efe10,
            0x1ab93d1d, 0x0ba5a4df, 0xa186f20f, 0x2868f169,
            0xdcb7da83, 0x573906fe, 0xa1e2ce9b, 0x4fcd7f52,
            0x50115e01, 0xa70683fa, 0xa002b5c4, 0x0de6d027,
            0x9af88c27, 0x773f8641, 0xc3604c06, 0x61a806b5,
            0xf0177a28, 0xc0f586e0, 0x006058aa, 0x30dc7d62,
            0x11e69ed7, 0x2338ea63, 0x53c2dd94, 0xc2c21634,
            0xbbcbee56, 0x90bcb6de, 0xebfc7da1, 0xce591d76,
            0x6f05e409, 0x4b7c0188, 0x39720a3d, 0x7c927c24,
            0x86e3725f, 0x724d9db9, 0x1ac15bb4, 0xd39eb8fc,
            0xed545578, 0x08fca5b5, 0xd83d7cd3, 0x4dad0fc4,
            0x1e50ef5e, 0xb161e6f8, 0xa28514d9, 0x6c51133c,
            0x6fd5c7e7, 0x56e14ec4, 0x362abfce, 0xddc6c837,
            0xd79a3234, 0x92638212, 0x670efa8e, 0x406000e0,
            0x3a39ce37, 0xd3faf5cf, 0xabc27737, 0x5ac52d1b,
            0x5cb0679e, 0x4fa33742, 0xd3822740, 0x99bc9bbe,
            0xd5118e9d, 0xbf0f7315, 0xd62d1c7e, 0xc700c47b,
            0xb78c1b6b, 0x21a19045, 0xb26eb1be, 0x6a366eb4,
            0x5748ab2f, 0xbc946e79, 0xc6a376d2, 0x6549c2c8,
            0x530ff8ee, 0x468dde7d, 0xd5730a1d, 0x4cd04dc6,
            0x2939bbdb, 0xa9ba4650, 0xac9526e8, 0xbe5ee304,
            0xa1fad5f0, 0x6a2d519a, 0x63ef8ce2, 0x9a86ee22,
            0xc089c2b8, 0x43242ef6, 0xa51e03aa, 0x9cf2d0a4,
            0x83c061ba, 0x9be96a4d, 0x8fe51550, 0xba645bd6,
            0x2826a2f9, 0xa73a3ae1, 0x4ba99586, 0xef5562e9,
            0xc72fefd3, 0xf752f7da, 0x3f046f69, 0x77fa0a59,
            0x80e4a915, 0x87b08601, 0x9b09e6ad, 0x3b3ee593,
            0xe990fd5a, 0x9e34d797, 0x2cf0b7d9, 0x022b8b51,
            0x96d5ac3a, 0x017da67d, 0xd1cf3ed6, 0x7c7d2d28,
            0x1f9f25cf, 0xadf2b89b, 0x5ad6b472, 0x5a88f54c,
            0xe029ac71, 0xe019a5e6, 0x47b0acfd, 0xed93fa9b,
            0xe8d3c48d, 0x283b57cc, 0xf8d56629, 0x79132e28,
            0x785f0191, 0xed756055, 0xf7960e44, 0xe3d35e8c,
            0x15056dd4, 0x88f46dba, 0x03a16125, 0x0564f0bd,
            0xc3eb9e15, 0x3c9057a2, 0x97271aec, 0xa93a072a,
            0x1b3f6d9b, 0x1e6321f5, 0xf59c66fb, 0x26dcf319,
            0x7533d928, 0xb155fdf5, 0x03563482, 0x8aba3cbb,
            0x28517711, 0xc20ad9f8, 0xabcc5167, 0xccad925f,
            0x4de81751, 0x3830dc8e, 0x379d5862, 0x9320f991,
            0xea7a90c2, 0xfb3e7bce, 0x5121ce64, 0x774fbe32,
            0xa8b6e37e, 0xc3293d46, 0x48de5369, 0x6413e680,
            0xa2ae0810, 0xdd6db224, 0x69852dfd, 0x09072166,
            0xb39a460a, 0x6445c0dd, 0x586cdecf, 0x1c20c8ae,
            0x5bbef7dd, 0x1b588d40, 0xccd2017f, 0x6bb4e3bb,
            0xdda26a7e, 0x3a59ff45, 0x3e350a44, 0xbcb4cdd5,
            0x72eacea8, 0xfa6484bb, 0x8d6612ae, 0xbf3c6f47,
            0xd29be463, 0x542f5d9e, 0xaec2771b, 0xf64e6370,
            0x740e0d8d, 0xe75b1357, 0xf8721671, 0xaf537d5d,
            0x4040cb08, 0x4eb4e2cc, 0x34d2466a, 0x0115af84,
            0xe1b00428, 0x95983a1d, 0x06b89fb4, 0xce6ea048,
            0x6f3f3b82, 0x3520ab82, 0x011a1d4b, 0x277227f8,
            0x611560b1, 0xe7933fdc, 0xbb3a792b, 0x344525bd,
            0xa08839e1, 0x51ce794b, 0x2f32c9b7, 0xa01fbac9,
            0xe01cc87e, 0xbcc7d1f6, 0xcf0111c3, 0xa1e8aac7,
            0x1a908749, 0xd44fbd9a, 0xd0dadecb, 0xd50ada38,
            0x0339c32a, 0xc6913667, 0x8df9317c, 0xe0b12b4f,
            0xf79e59b7, 0x43f5bb3a, 0xf2d519ff, 0x27d9459c,
            0xbf97222c, 0x15e6fc2a, 0x0f91fc71, 0x9b941525,
            0xfae59361, 0xceb69ceb, 0xc2a86459, 0x12baa8d1,
            0xb6c1075e, 0xe3056a0c, 0x10d25065, 0xcb03a442,
            0xe0ec6e0e, 0x1698db3b, 0x4c98a0be, 0x3278e964,
            0x9f1f9532, 0xe0d392df, 0xd3a0342b, 0x8971f21e,
            0x1b0a7441, 0x4ba3348c, 0xc5be7120, 0xc37632d8,
            0xdf359f8d, 0x9b992f2e, 0xe60b6f47, 0x0fe3f11d,
            0xe54cda54, 0x1edad891, 0xce6279cf, 0xcd3e7e6f,
            0x1618b166, 0xfd2c1d05, 0x848fd2c5, 0xf6fb2299,
            0xf523f357, 0xa6327623, 0x93a83531, 0x56cccd02,
            0xacf08162, 0x5a75ebb5, 0x6e163697, 0x88d273cc,
            0xde966292, 0x81b949d0, 0x4c50901b, 0x71c65614,
            0xe6c6c7bd, 0x327a140a, 0x45e1d006, 0xc3f27b9a,
            0xc9aa53fd, 0x62a80f00, 0xbb25bfe2, 0x35bdd2f6,
            0x71126905, 0xb2040222, 0xb6cbcf7c, 0xcd769c2b,
            0x53113ec0, 0x1640e3d3, 0x38abbd60, 0x2547adf0,
            0xba38209c, 0xf746ce76, 0x77afa1c5, 0x20756060,
            0x85cbfe4e, 0x8ae88dd8, 0x7aaaf9b0, 0x4cf9aa7e,
            0x1948c25c, 0x02fb8a8c, 0x01c36ae4, 0xd6ebe1f9,
            0x90d4f869, 0xa65cdea0, 0x3f09252d, 0xc208e69f,
            0xb74e6132, 0xce77e25b, 0x578fdfe3, 0x3ac372e6
    };

    // bcrypt IV: "OrpheanBeholderScryDoubt". The C implementation calls
    // this "ciphertext", but it is really plaintext or an IV. We keep
    // the name to make code comparison easier.
    static private final int bf_crypt_ciphertext[] = {
            0x4f727068, 0x65616e42, 0x65686f6c,
            0x64657253, 0x63727944, 0x6f756274
    };

    // Table for Base64 encoding
    static private final char base64_code[] = {
            '.', '/', 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J',
            'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V',
            'W', 'X', 'Y', 'Z', 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h',
            'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't',
            'u', 'v', 'w', 'x', 'y', 'z', '0', '1', '2', '3', '4', '5',
            '6', '7', '8', '9'
    };

    // Table for Base64 decoding
    static private final byte index_64[] = {
            -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
            -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
            -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
            -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
            -1, -1, -1, -1, -1, -1, 0, 1, 54, 55,
            56, 57, 58, 59, 60, 61, 62, 63, -1, -1,
            -1, -1, -1, -1, -1, 2, 3, 4, 5, 6,
            7, 8, 9, 10, 11, 12, 13, 14, 15, 16,
            17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27,
            -1, -1, -1, -1, -1, -1, 28, 29, 30,
            31, 32, 33, 34, 35, 36, 37, 38, 39, 40,
            41, 42, 43, 44, 45, 46, 47, 48, 49, 50,
            51, 52, 53, -1, -1, -1, -1, -1
    };

    // Expanded Blowfish key
    private int P[];
    private int S[];

    /**
     * Encode a byte array using bcrypt's slightly-modified base64
     * encoding scheme. Note that this is *not* compatible with
```

```
     * the standard MIME-base64 encoding.
     *
     * @param d the byte array to encode
     * @param len   the number of bytes to encode
     * @return  base64-encoded string
     * @exception IllegalArgumentException if the length is invalid
     */
    private static String encode_base64(byte d[], int len)
            throws IllegalArgumentException {
        int off = 0;
        StringBuffer rs = new StringBuffer();
        int c1, c2;

        if (len <= 0 || len > d.length) {
            throw new IllegalArgumentException("Invalid len");
        }
        while (off < len) {
            c1 = d[off++] & 0xff;
            rs.append(base64_code[(c1 >> 2) & 0x3f]);
            c1 = (c1 & 0x03) << 4;
            if (off >= len) {
                rs.append(base64_code[c1 & 0x3f]);
                break;
            }
            c2 = d[off++] & 0xff;
            c1 |= (c2 >> 4) & 0x0f;
            rs.append(base64_code[c1 & 0x3f]);
            c1 = (c2 & 0x0f) << 2;
            if (off >= len) {
                rs.append(base64_code[c1 & 0x3f]);
                break;
            }
            c2 = d[off++] & 0xff;
            c1 |= (c2 >> 6) & 0x03;
            rs.append(base64_code[c1 & 0x3f]);
            rs.append(base64_code[c2 & 0x3f]);
        }
        return rs.toString();
    }

    /**
     * Look up the 3 bits base64-encoded by the specified character,
     * range-checking againt conversion table
     * @param x the base64-encoded value
     * @return  the decoded value of x
     */
    private static byte char64(char x) {
        if ((int)x < 0 || (int)x > index_64.length) {
            return -1;
        }
        return index_64[(int)x];
    }

    /**
     * Decode a string encoded using bcrypt's base64 scheme to a
     * byte array. Note that this is *not* compatible with
     * the standard MIME-base64 encoding.
     * @param s the string to decode
     * @param maxolen   the maximum number of bytes to decode
     * @return  an array containing the decoded bytes
     * @throws IllegalArgumentException if maxolen is invalid
     */
    private static byte[] decode_base64(String s, int maxolen)
            throws IllegalArgumentException {
        StringBuffer rs = new StringBuffer();
        int off = 0, slen = s.length(), olen = 0;
        byte ret[];
        byte c1, c2, c3, c4, o;

        if (maxolen <= 0) {
            throw new IllegalArgumentException("Invalid maxolen");
        }
        while (off < slen - 1 && olen < maxolen) {
            c1 = char64(s.charAt(off++));
            c2 = char64(s.charAt(off++));
            if (c1 == -1 || c2 == -1) {
                break;
            }
            o = (byte)(c1 << 2);
            o |= (c2 & 0x30) >> 4;
            rs.append((char)o);
            if (++olen >= maxolen || off >= slen) {
                break;
            }
            c3 = char64(s.charAt(off++));
            if (c3 == -1) {
                break;
            }
            o = (byte)((c2 & 0x0f) << 4);
            o |= (c3 & 0x3c) >> 2;
            rs.append((char)o);
            if (++olen >= maxolen || off >= slen) {
                break;
            }
            c4 = char64(s.charAt(off++));
            o = (byte)((c3 & 0x03) << 6);
            o |= c4;
            rs.append((char)o);
            ++olen;
        }

        ret = new byte[olen];
        for (off = 0; off < olen; off++) {
            ret[off] = (byte) rs.charAt(off);
        }
        return ret;
    }

    /**
     * Blowfish encipher a single 64-bit block encoded as
     * two 32-bit halves
     * @param lr    an array containing the two 32-bit half blocks
     * @param off   the position in the array of the blocks
     */
    private final void encipher(int lr[], int off) {
        int i, n, l = lr[off], r = lr[off + 1];
        l ^= P[0];
        for (i = 0; i <= BLOWFISH_NUM_ROUNDS - 2;) {
            // Feistel substitution on left word
            n = S[(l >> 24) & 0xff];
            n += S[0x100 | ((l >> 16) & 0xff)];
            n ^= S[0x200 | ((l >> 8) & 0xff)];
            n += S[0x300 | (l & 0xff)];
            r ^= n ^ P[++i];

            // Feistel substitution on right word
            n = S[(r >> 24) & 0xff];
            n += S[0x100 | ((r >> 16) & 0xff)];
            n ^= S[0x200 | ((r >> 8) & 0xff)];
            n += S[0x300 | (r & 0xff)];
            l ^= n ^ P[++i];
        }
        lr[off] = r ^ P[BLOWFISH_NUM_ROUNDS + 1];
        lr[off + 1] = l;
    }

    /**
     * Cycically extract a word of key material
     * @param data  the string to extract the data from
     * @param offp  a "pointer" (as a one-entry array) to the
     * current offset into data
     * @return  the next word of material from data
     */
    private static int streamtoword(byte data[], int offp[]) {
        int i;
        int word = 0;
        int off = offp[0];

        for (i = 0; i < 4; i++) {
            word = (word << 8) | (data[off] & 0xff);
            off = (off + 1) % data.length;
        }

        offp[0] = off;
        return word;
    }

    /**
     * Initialise the Blowfish key schedule
     */
    private void init_key() {
        P = (int[])P_orig.clone();
        S = (int[])S_orig.clone();
    }

    /**
     * Key the Blowfish cipher
     * @param key   an array containing the key
     */
    private void key(byte key[]) {
        int i;
        int koffp[] = { 0 };
        int lr[] = { 0, 0 };
        int plen = P.length, slen = S.length;

        for (i = 0; i < plen; i++) {
            P[i] = P[i] ^ streamtoword(key, koffp);
        }
        for (i = 0; i < plen; i += 2) {
            encipher(lr, 0);
            P[i] = lr[0];
            P[i + 1] = lr[1];
        }

        for (i = 0; i < slen; i += 2) {
            encipher(lr, 0);
            S[i] = lr[0];
            S[i + 1] = lr[1];
        }
    }

    /**
     * Perform the "enhanced key schedule" step described by
     * Provos and Mazieres in "A Future-Adaptable Password Scheme"
     * http://www.openbsd.org/papers/bcrypt-paper.ps
     * @param data  salt information
     * @param key   password information
     */
    private void ekskey(byte data[], byte key[]) {
        int i;
        int koffp[] = { 0 }, doffp[] = { 0 };
        int lr[] = { 0, 0 };
        int plen = P.length, slen = S.length;

        for (i = 0; i < plen; i++) {
            P[i] = P[i] ^ streamtoword(key, koffp);
        }
        for (i = 0; i < plen; i += 2) {
            lr[0] ^= streamtoword(data, doffp);
            lr[1] ^= streamtoword(data, doffp);
            encipher(lr, 0);
            P[i] = lr[0];
            P[i + 1] = lr[1];
        }

        for (i = 0; i < slen; i += 2) {
            lr[0] ^= streamtoword(data, doffp);
            lr[1] ^= streamtoword(data, doffp);
            encipher(lr, 0);
            S[i] = lr[0];
            S[i + 1] = lr[1];
        }
    }

    /**
     * Perform the central password hashing step in the
     * bcrypt scheme
     * @param password  the password to hash
     * @param salt  the binary salt to hash with the password
     * @param log_rounds    the binary logarithm of the number
     * of rounds of hashing to apply
```

```java
     * @param cdata         the plaintext to encrypt
     * @return  an array containing the binary hashed password
     */
    public byte[] crypt_raw(byte password[], byte salt[], int log_rounds,
                            int cdata[]) {
        int rounds, i, j;
        int clen = cdata.length;
        byte ret[];

        if (log_rounds < 4 || log_rounds > 30) {
            throw new IllegalArgumentException("Bad number of rounds");
        }
        rounds = 1 << log_rounds;
        if (salt.length != BCRYPT_SALT_LEN) {
            throw new IllegalArgumentException("Bad salt length");
        }
        init_key();
        ekskey(salt, password);
        for (i = 0; i != rounds; i++) {
            key(password);
            key(salt);
        }

        for (i = 0; i < 64; i++) {
            for (j = 0; j < (clen >> 1); j++) {
                encipher(cdata, j << 1);
            }
        }

        ret = new byte[clen * 4];
        for (i = 0, j = 0; i < clen; i++) {
            ret[j++] = (byte)((cdata[i] >> 24) & 0xff);
            ret[j++] = (byte)((cdata[i] >> 16) & 0xff);
            ret[j++] = (byte)((cdata[i] >> 8) & 0xff);
            ret[j++] = (byte)(cdata[i] & 0xff);
        }
        return ret;
    }

    /**
     * Hash a password using the OpenBSD bcrypt scheme
     * @param password  the password to hash
     * @param salt  the salt to hash with (perhaps generated
     * using BCrypt.gensalt)
     * @return  the hashed password
     */
    public static String hashpw(String password, String salt) {
        BCryptUtil B;
        String real_salt;
        byte passwordb[], saltb[], hashed[];
        char minor = (char)0;
        int rounds, off = 0;
        StringBuffer rs = new StringBuffer();

        if (salt.charAt(0) != '$' || salt.charAt(1) != '2') {
            throw new IllegalArgumentException("Invalid salt version");
        }
        if (salt.charAt(2) == '$') {
            off = 3;
        } else {
            minor = salt.charAt(2);
            if (minor != 'a' || salt.charAt(3) != '$') {
                throw new IllegalArgumentException("Invalid salt revision");
            }
            off = 4;
        }

        // Extract number of rounds
        if (salt.charAt(off + 2) > '$') {
            throw new IllegalArgumentException("Missing salt rounds");
        }
        rounds = Integer.parseInt(salt.substring(off, off + 2));

        real_salt = salt.substring(off + 3, off + 25);
        try {
            passwordb = (password + (minor >= 'a' ? "\000" : "")).getBytes("UTF-
8");
        } catch (UnsupportedEncodingException uee) {
            throw new AssertionError("UTF-8 is not supported");
        }

        saltb = decode_base64(real_salt, BCRYPT_SALT_LEN);

        B = new BCryptUtil();
        hashed = B.crypt_raw(passwordb, saltb, rounds,
                (int[])bf_crypt_ciphertext.clone());

        rs.append("$2");
        if (minor >= 'a') {
            rs.append(minor);
        }
        rs.append("$");
        if (rounds < 10) {
            rs.append("0");
        }
        if (rounds > 30) {
            throw new IllegalArgumentException(
                    "rounds exceeds maximum (30)");
        }
        rs.append(Integer.toString(rounds));
        rs.append("$");
        rs.append(encode_base64(saltb, saltb.length));
        rs.append(encode_base64(hashed,
                bf_crypt_ciphertext.length * 4 - 1));
        return rs.toString();
    }

    /**
     * Generate a salt for use with the BCrypt.hashpw() method
     * @param log_rounds    the log2 of the number of rounds of
     * hashing to apply - the work factor therefore increases as
     * 2**log_rounds.
     * @param random        an instance of SecureRandom to use
     * @return  an encoded salt value
     */
    public static String gensalt(int log_rounds, SecureRandom random) {
        StringBuffer rs = new StringBuffer();
        byte rnd[] = new byte[BCRYPT_SALT_LEN];

        random.nextBytes(rnd);

        rs.append("$2a$");
        if (log_rounds < 10) {
            rs.append("0");
        }
        if (log_rounds > 30) {
            throw new IllegalArgumentException(
                    "log_rounds exceeds maximum (30)");
        }
        rs.append(Integer.toString(log_rounds));
        rs.append("$");
        rs.append(encode_base64(rnd, rnd.length));
        return rs.toString();
    }

    /**
     * Generate a salt for use with the BCrypt.hashpw() method
     * @param log_rounds    the log2 of the number of rounds of
     * hashing to apply - the work factor therefore increases as
     * 2**log_rounds.
     * @return  an encoded salt value
     */
    public static String gensalt(int log_rounds) {
        return gensalt(log_rounds, new SecureRandom());
    }

    /**
     * Generate a salt for use with the BCrypt.hashpw() method,
     * selecting a reasonable default for the number of hashing
     * rounds to apply
     * @return  an encoded salt value
     */
    public static String gensalt() {
        return gensalt(GENSALT_DEFAULT_LOG2_ROUNDS);
    }

    /**
     * Check that a plaintext password matches a previously hashed
     * one
     * @param plaintext the plaintext password to verify
     * @param hashed    the previously-hashed password
     * @return  true if the passwords match, false otherwise
     */
    public static boolean checkpw(String plaintext, String hashed) {
        byte hashed_bytes[];
        byte try_bytes[];
        try {
            String try_pw = hashpw(plaintext, hashed);
            hashed_bytes = hashed.getBytes("UTF-8");
            try_bytes = try_pw.getBytes("UTF-8");
        } catch (UnsupportedEncodingException uee) {
            return false;
        }
        if (hashed_bytes.length != try_bytes.length) {
            return false;
        }
        byte ret = 0;
        for (int i = 0; i < try_bytes.length; i++) {
            ret |= hashed_bytes[i] ^ try_bytes[i];
        }
        return ret == 0;
    }

/*    public static void main(String[] args) {
        String password = "123456";
        String mingruijiaoyu = hashpw(password, gensalt());//加密
        System.out.println(mingruijiaoyu);

        String pwd =
"$2a$10$nA2g16Zq1xS.CinETDlKfuxqLbZWMgIVSCGsE7G2G27PicxqoKsL.";

        boolean checkpw = checkpw("1234556", pwd);//明文密码和密文密码比较
        System.out.println(checkpw);
    }*/
}
```

2.2.2 bctrypt特点

- 每一次 HASH 出来的值不一样
- 计算非常缓慢

因此使用 Bcrypt 进行加密后，攻击者想要使用算出 M2 成本变得不可接受。但代价是应用自身也会性能受到影响，不过登录行为并不是随时在发生，因此能够忍受。对于攻击者来说，需要不断计算，让攻击变得不太可能。

spring-security就使用了bctrypt的加密算法

https://github.com/spring-projects/spring-security/blob/master/crypto/src/main/java/org/springframework/security/crypto/bcrypt/BCrypt.java

我们当前的工具类只是将spring官方提供的工具类原封不动的复制了出来,改了一个类名而已.....

2.3 service

2.3.1 新建mingrui-shop-service-user项目

2.3.2 pom.xml

```
    <dependencies>
        <dependency>
            <groupId>com.baidu</groupId>
            <artifactId>mingrui-shop-service-api-user</artifactId>
            <version>1.0-SNAPSHOT</version>
        </dependency>
    </dependencies>
```

2.3.3 application.yml

```
server:
  port: 8500

spring:
  application:
    name: user-server
  datasource:
    # 数据源名称，任意
    name: mysql
    url: jdbc:mysql://127.0.0.1:3306/mr-shop?
useSSL=true&nullNamePatternMatchesAll=true&serverTimezone=GMT%2B8&useUnicode=tru
e&characterEncoding=utf8
    # 数据库连接用户
    username: root
    # 数据库连接密码
    password: root
    # 驱动名称
    driver-class-name: com.mysql.cj.jdbc.Driver
    # boot2.0+使用hikari作为默认数据库连接池
    type: com.zaxxer.hikari.HikariDataSource
    hikari:
      # 是否自动提交事务 默认
      auto-commit: true
      # 允许的最小连接数
      minimum-idle: 5
      # 连接池内最大连接数
      maximum-pool-size: 10
      # 验证连接的sql语句
      connection-test-query: SELECT 1 FROM DUAL
      # 连接超时时间 默认30000 毫秒 如果小于250毫秒，则被重置回30秒
      connection-timeout: 30000
      # 验证超时时间默认5000毫秒 如果小于250毫秒，则会被重置回5秒
      validation-timeout: 5000
      # 设置连接在连接池中的存货时间 如果不等于0且小于30秒则会被重置回30分钟
      max-lifetime: 1800000
# 通用mapper
mapper:
  mappers: tk.mybatis.mapper.common.Mapper
  identity: MYSQL
#日志设置
logging:
  level:
    # 打印与我们程序相关的日志信息
    com.baidu.shop: debug
# eureka配置
eureka:
  client:
    service-url:
      defaultZone: http://localhost:8761/eureka/
```

2.3.4 新建包com.baidu

2.3.5 新建启动类RunUserServerApplication

```
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
```

```java
import org.springframework.cloud.netflix.eureka.EnableEurekaClient;
import tk.mybatis.spring.annotation.MapperScan;

/**
 * @ClassName RunUserServerApplication
 * @Description: TODO
 * @Author shenyaqi
 * @Date 2020/9/20
 * @Version V1.0
 **/
@SpringBootApplication
@EnableEurekaClient
@MapperScan(value = "com.baidu.shop.mapper")
public class RunUserServerApplication {
    public static void main(String[] args) {
        SpringApplication.run(RunUserServerApplication.class);
    }
}
```

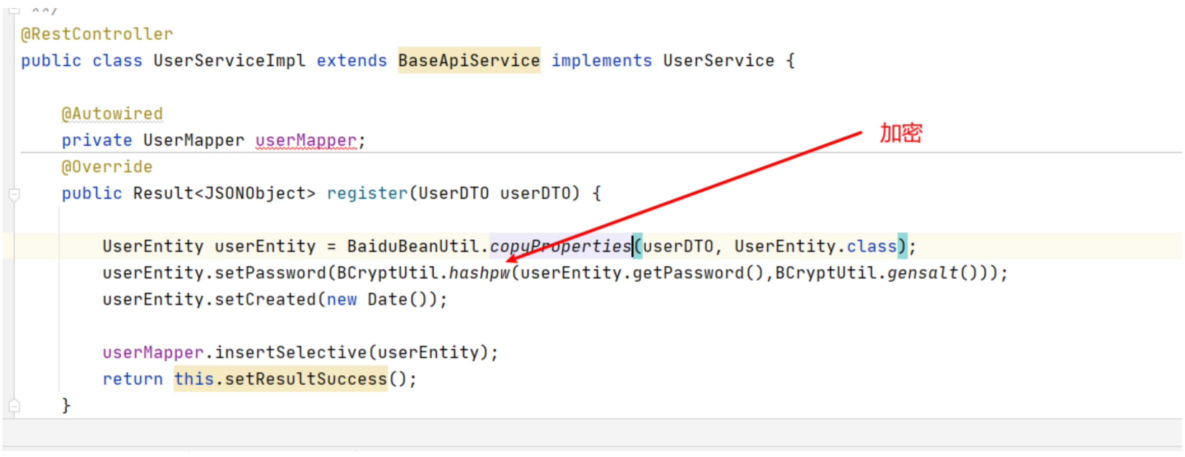
2.3.6 新建包com.baidu.shop.mapper

2.3.7 包下新建UserMapper

```java
import com.baidu.shop.entity.UserEntity;
import tk.mybatis.mapper.common.Mapper;

/**
 * @ClassName UserMapper
 * @Description: TODO
 * @Author shenyaqi
 * @Date 2020/9/20
 * @Version V1.0
 **/
public interface UserMapper extends Mapper<UserEntity> {
}
```

2.3.8 新建包com.baidu.shop.service.impl

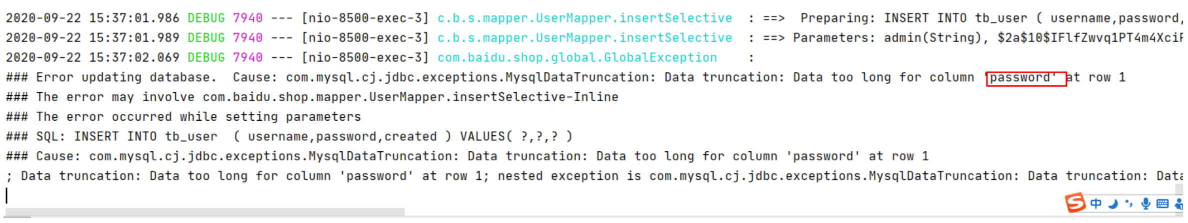
2.3.9 包下新建UserServiceImpl



```java
@RestController
public class UserServiceImpl extends BaseApiService implements UserService {

    @Autowired
    private UserMapper userMapper;
    @Override
    public Result<JSONObject> register(UserDTO userDTO) {

        UserEntity userEntity = BaiduBeanUtil.copyProperties(userDTO, UserEntity.class);
        userEntity.setPassword(BCryptUtil.hashpw(userEntity.getPassword(),BCryptUtil.gensalt()));
        userEntity.setCreated(new Date());

        userMapper.insertSelective(userEntity);
        return this.setResultSuccess();
    }
```

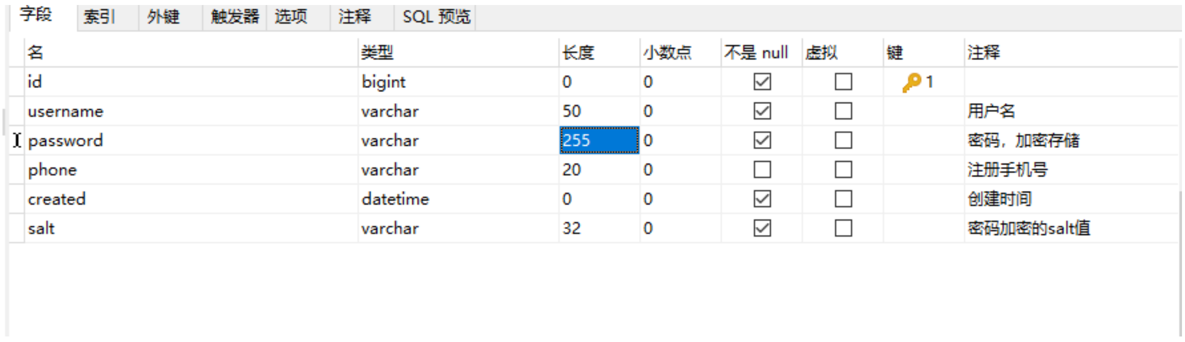
```java
import com.alibaba.fastjson.JSONObject;
import com.baidu.shop.base.BaseApiService;
import com.baidu.shop.base.Result;
import com.baidu.shop.dto.UserDTO;
import com.baidu.shop.entity.UserEntity;
import com.baidu.shop.mapper.UserMapper;
import com.baidu.shop.service.UserService;
import com.baidu.shop.utils.BCryptUtil;
import com.baidu.shop.utils.BaiduBeanUtil;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.RestController;

import java.util.Date;

/**
 * @ClassName UserServiceImpl
 * @Description: TODO
 * @Author shenyaqi
 * @Date 2020/9/20
 * @Version V1.0
 **/
@RestController
public class UserServiceImpl extends BaseApiService implements UserService {

    @Autowired
    private UserMapper userMapper;
    @Override
    public Result<JSONObject> register(UserDTO userDTO) {

        UserEntity userEntity = BaiduBeanUtil.copyProperties(userDTO,
UserEntity.class);

 userEntity.setPassword(BCryptUtil.hashpw(userEntity.getPassword(),BCryptUtil.ge
nsalt()));
        userEntity.setCreated(new Date());

        userMapper.insertSelective(userEntity);
        return this.setResultSuccess();
    }
}
```

2.3.10 测试注册接口

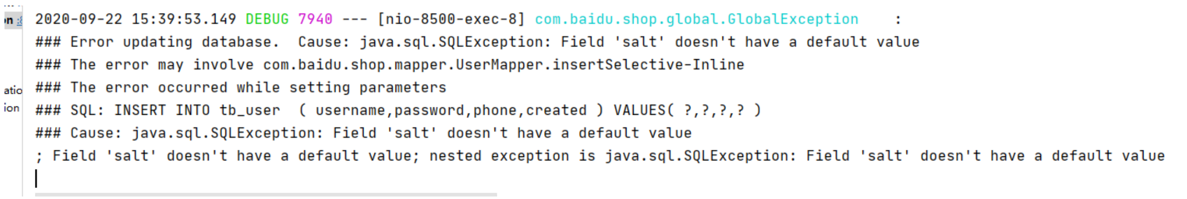


```
2020-09-22 15:37:01.986 DEBUG 7940 --- [nio-8500-exec-3] c.b.s.mapper.UserMapper.insertSelective  : ==>  Preparing: INSERT INTO tb_user ( username,password,
2020-09-22 15:37:01.989 DEBUG 7940 --- [nio-8500-exec-3] c.b.s.mapper.UserMapper.insertSelective  : ==> Parameters: admin(String), $2a$10$IFlfZwvq1PT4m4XciF
2020-09-22 15:37:02.069 DEBUG 7940 --- [nio-8500-exec-3] com.baidu.shop.global.GlobalException    :
### Error updating database.  Cause: com.mysql.cj.jdbc.exceptions.MysqlDataTruncation: Data truncation: Data too long for column 'password' at row 1
### The error may involve com.baidu.shop.mapper.UserMapper.insertSelective-Inline
### The error occurred while setting parameters
### SQL: INSERT INTO tb_user  ( username,password,created ) VALUES( ?,?,? )
### Cause: com.mysql.cj.jdbc.exceptions.MysqlDataTruncation: Data truncation: Data too long for column 'password' at row 1
; Data truncation: Data too long for column 'password' at row 1; nested exception is com.mysql.cj.jdbc.exceptions.MysqlDataTruncation: Data truncation: Data
```

数据库表的密码长度给的不够

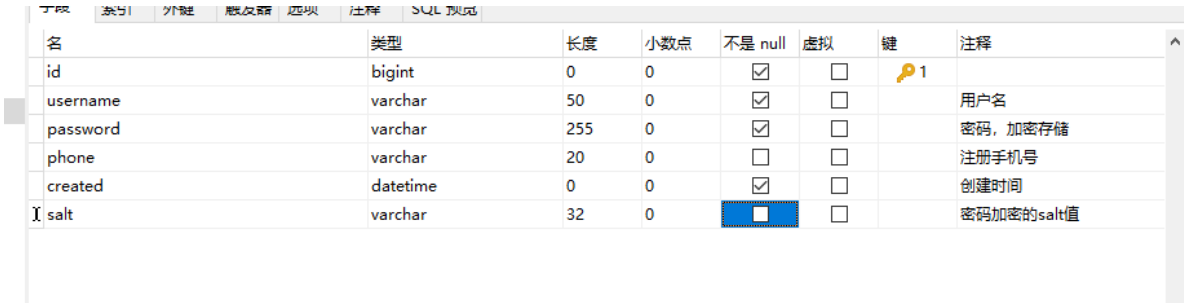


名	类型	长度	小数点	不是 null	虚拟	键	注释
id	bigint	0	0	☑	☐	🔑1	
username	varchar	50	0	☑	☐		用户名
password	varchar	255	0	☑	☐		密码，加密存储
phone	varchar	20	0	☐	☐		注册手机号
created	datetime	0	0	☑	☐		创建时间
salt	varchar	32	0	☑	☐		密码加密的salt值



```
2020-09-22 15:39:53.149 DEBUG 7940 --- [nio-8500-exec-8] com.baidu.shop.global.GlobalException    :
### Error updating database.  Cause: java.sql.SQLException: Field 'salt' doesn't have a default value
### The error may involve com.baidu.shop.mapper.UserMapper.insertSelective-Inline
### The error occurred while setting parameters
### SQL: INSERT INTO tb_user  ( username,password,phone,created ) VALUES( ?,?,?,? )
### Cause: java.sql.SQLException: Field 'salt' doesn't have a default value
; Field 'salt' doesn't have a default value; nested exception is java.sql.SQLException: Field 'salt' doesn't have a default value
```

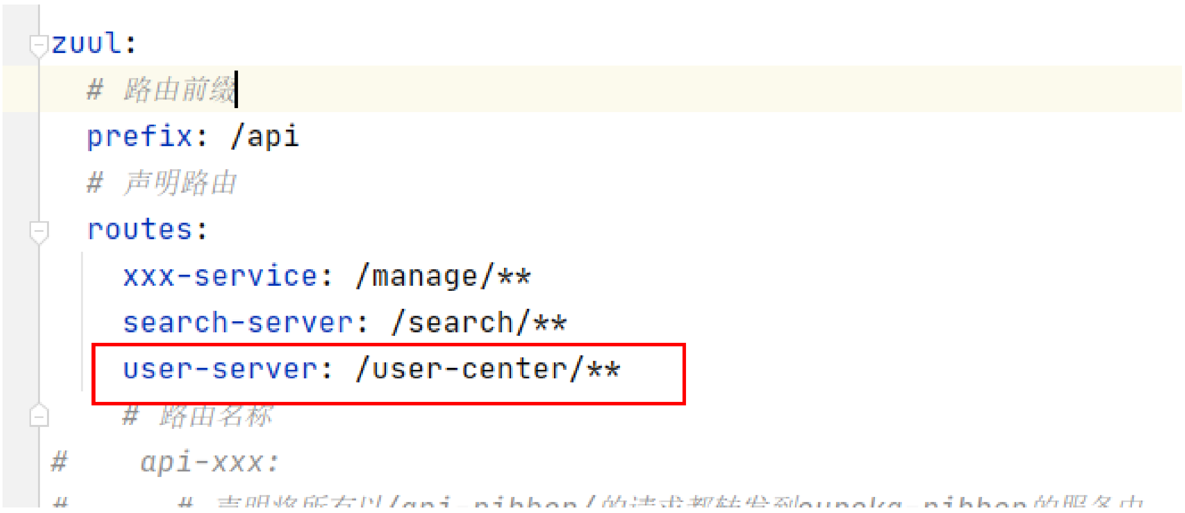
salt这个字段我们没有用,但是数据库设置的这个字段不能为空



名	类型	长度	小数点	不是 null	虚拟	键	注释
id	bigint	0	0	☑	☐	🔑1	
username	varchar	50	0	☑	☐		用户名
password	varchar	255	0	☑	☐		密码，加密存储
phone	varchar	20	0	☐	☐		注册手机号
created	datetime	0	0	☑	☐		创建时间
salt	varchar	32	0	☐	☐		密码加密的salt值

2.4 zuul

2.4.1 application.yml



```
zuul:
  # 路由前缀
  prefix: /api
  # 声明路由
  routes:
    xxx-service: /manage/**
    search-server: /search/**
    user-server: /user-center/**
    # 路由名称
#     api-xxx:
#       # 声明将所有以/api-ribbon/的请求都转发到eureka-ribbon的服务中
```

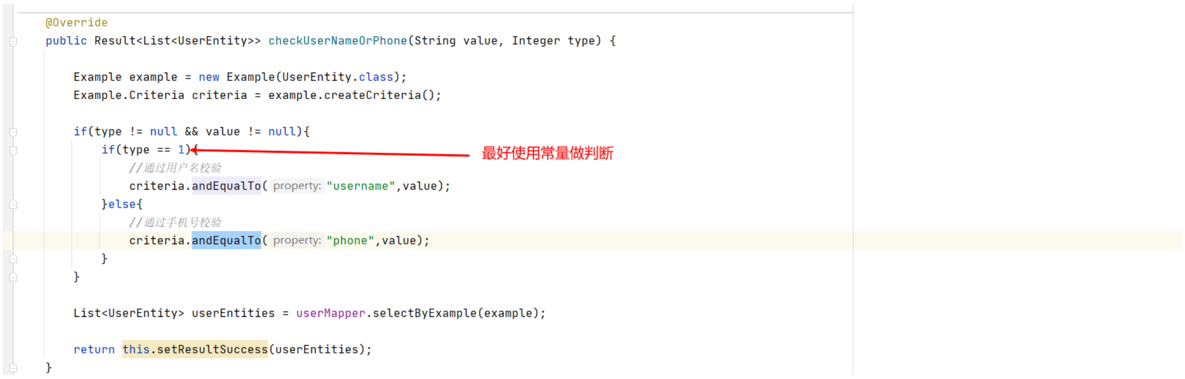
3 注册功能实现

3.1 校验用户名/手机号唯一

3.1.1 UserService

```
    @ApiOperation(value = "校验用户名或手机号唯一")
    @GetMapping(value = "user/check/{value}/{type}")
    Result<List<UserEntity>> checkUserNameOrPhone(@PathVariable(value = "value")
String value, @PathVariable(value = "type") Integer type);
```

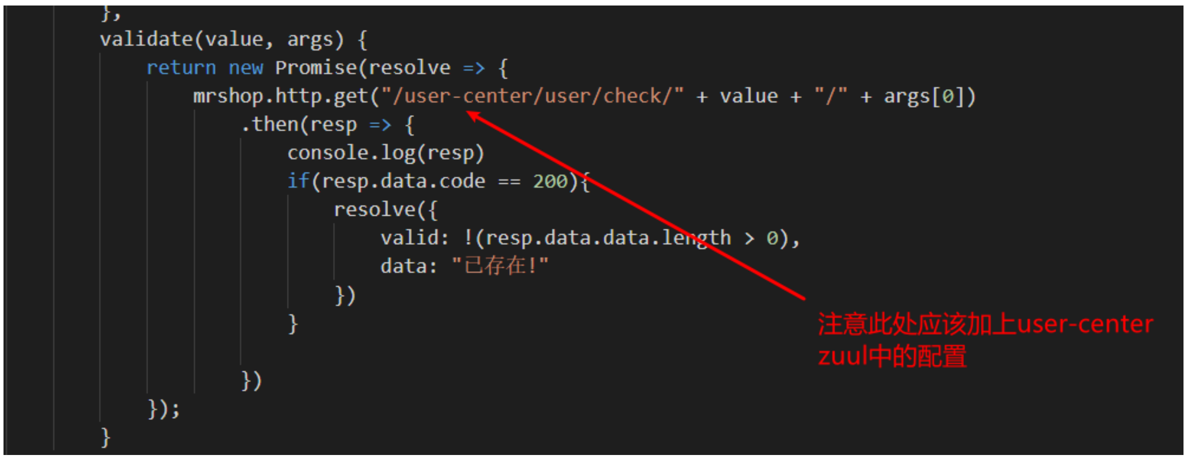
3.1.2 UserServiceImpl



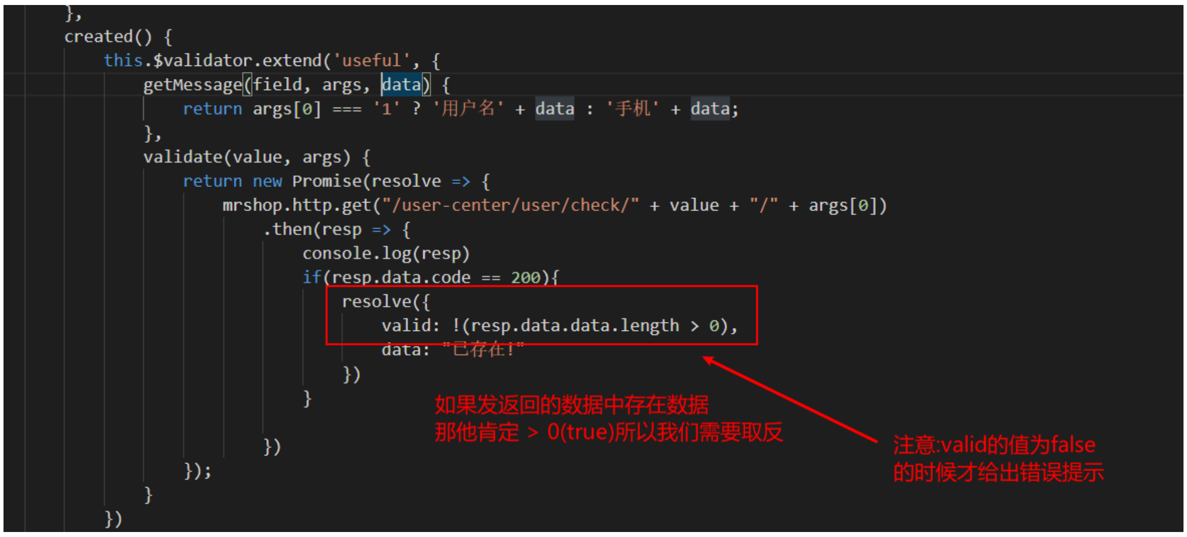
```
@Override
public Result<List<UserEntity>> checkUserNameOrPhone(String value, Integer type) {

    Example example = new Example(UserEntity.class);
    Example.Criteria criteria = example.createCriteria();

    if(type != null && value != null){
        if(type == 1){
            //通过用户名校验
            criteria.andEqualTo( property: "username",value);
        }else{
            //通过手机号校验
            criteria.andEqualTo( property: "phone",value);
        }
    }

    List<UserEntity> userEntities = userMapper.selectByExample(example);

    return this.setResultSuccess(userEntities);
}
```

```
    @Override
    public Result<List<UserEntity>> checkUserNameOrPhone(String value, Integer
type) {

        Example example = new Example(UserEntity.class);
        Example.Criteria criteria = example.createCriteria();

        if(type != null && value != null){
            if(type == 1){
                //通过用户名校验
                criteria.andEqualTo("username",value);
            }else{
                //通过手机号校验
                criteria.andEqualTo("phone",value);
            }
        }

        List<UserEntity> userEntities = userMapper.selectByExample(example);

        return this.setResultSuccess(userEntities);
    }
```

页面代码



```
    },
    validate(value, args) {
        return new Promise(resolve => {
            mrshop.http.get("/user-center/user/check/" + value + "/" + args[0])
                .then(resp => {
                    console.log(resp)
                    if(resp.data.code == 200){
                        resolve({
                            valid: !(resp.data.data.length > 0),
                            data: "已存在!"
                        })
                    }

                })
        });
    }
```



```
    },
    created() {
        this.$validator.extend('useful', {
            getMessage(field, args, data) {
                return args[0] === '1' ? '用户名' + data : '手机' + data;
            },
            validate(value, args) {
                return new Promise(resolve => {
                    mrshop.http.get("/user-center/user/check/" + value + "/" + args[0])
                        .then(resp => {
                            console.log(resp)
                            if(resp.data.code == 200){
                                resolve({
                                    valid: !(resp.data.data.length > 0),
                                    data: "已存在!"
                                })
                            }

                        })
                });
            }
        })
```

如果发返回的数据中存在数据
那他肯定 > 0(true)所以我们需要取反

注意:valid的值为false
的时候才给出错误提示

```
                  validate(value, args) {
                      return new Promise(resolve => {
                          mrshop.http.get("/user-center/user/check/" + value + "/"
+ args[0])
                              .then(resp => {
                                  console.log(resp)
                                  if(resp.data.code == 200){
                                      resolve({
                                          valid: !(resp.data.data.length > 0),
                                          data: "已存在!"
                                      })
                                  }

                              })
                      });
                  }
```

3.2 给手机发送验证码

3.2.1 螺丝帽

文档：项目接入短信平台（螺丝帽）.note
链接：http://note.youdao.com/noteshare?id=423762507f2d04dd10c936498bb678e0&sub=9E5C9B6C0C6543428DE728EBA964E6CC

3.2.2 将本地jar包安装到本地仓库

```
mvn install:install-file -Dfile=jersey-bundle-1.19.jar -
DgroupId=com.mrshop.luosimao -DartifactId=jersey-bundle -Dversion=1.0.0 -
Dpackaging=jar

mvn install:install-file -Dfile=json-org.jar -DgroupId=com.mrshop.luosimao -
DartifactId=json-org -Dversion=1.0.0 -Dpackaging=jar
```

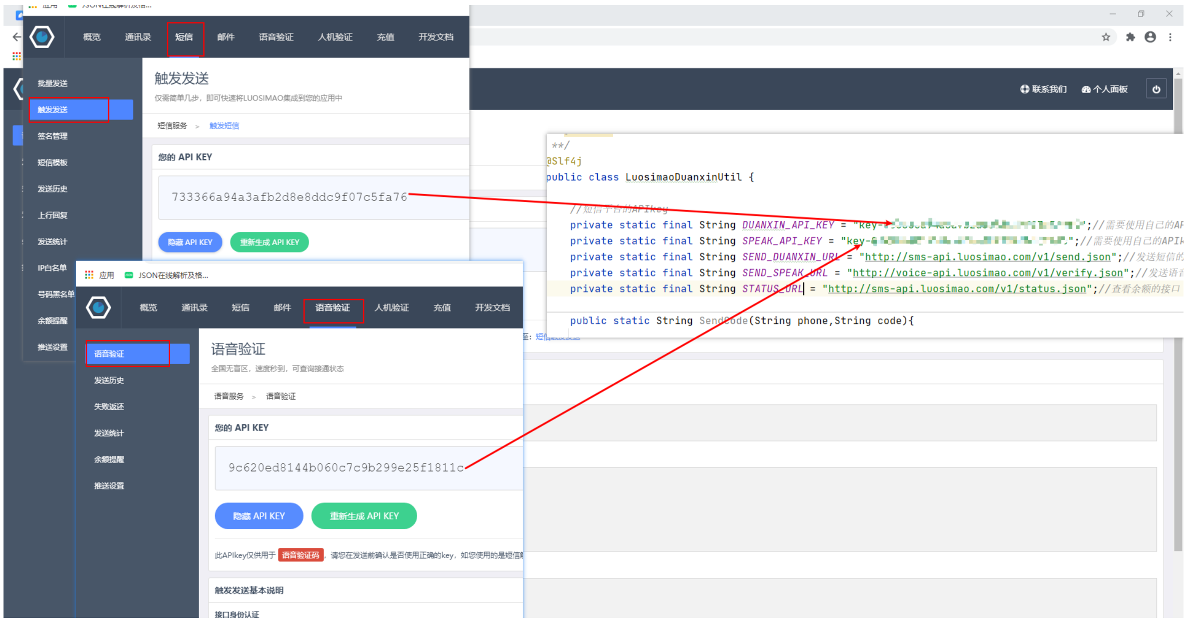
这么做不对,应该将jar包上传到maven私服

3.2.3 common-core

3.2.3.1 pom.xml

```xml
        <dependency>
            <groupId>com.mrshop.luosimao</groupId>
            <artifactId>jersey-bundle</artifactId>
            <version>1.0.0</version>
        </dependency>

        <dependency>
            <groupId>com.mrshop.luosimao</groupId>
            <artifactId>json-org</artifactId>
            <version>1.0.0</version>
        </dependency>
```

3.2.3.2 utils包下新建LuosimaoDuanxinUtil



触发发送

您的 API KEY

733366a94a3afb2d8e8ddc9f07c5fa76

语音验证

您的 API KEY

9c620ed8144b060c7c9b299e25f1811c

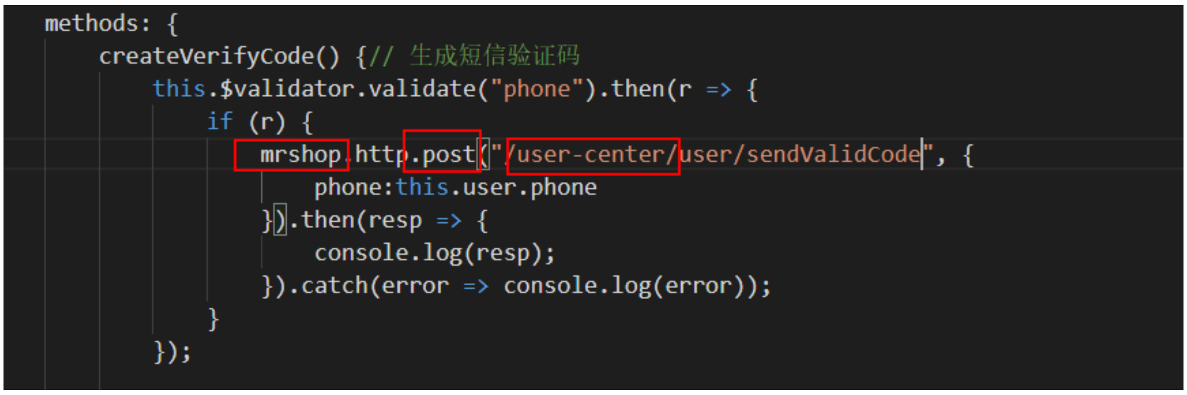
此APIkey仅供用于 语音验证码，请您在发送前确认是否使用正确的key，如您使用的是短信

```java
 **/
@Slf4j
public class LuosimaoDuanxinUtil {

    //短信平台的APIkey
    private static final String DUANXIN_API_KEY = "key-[redacted]";//需要使用自己的AP
    private static final String SPEAK_API_KEY = "key-[redacted]";//需要使用自己的APIk
    private static final String SEND_DUANXIN_URL = "http://sms-api.luosimao.com/v1/send.json";//发送短信的
    private static final String SEND_SPEAK_URL = "http://voice-api.luosimao.com/v1/verify.json";//发送语
    private static final String STATUS_URL = "http://sms-api.luosimao.com/v1/status.json";//查看余额的接口

    public static String SendCode(String phone,String code){
```

```java
import com.sun.jersey.api.client.Client;
import com.sun.jersey.api.client.ClientResponse;
import com.sun.jersey.api.client.WebResource;
import com.sun.jersey.api.client.filter.HTTPBasicAuthFilter;
import com.sun.jersey.core.util.MultivaluedMapImpl;
import lombok.extern.slf4j.Slf4j;
import javax.ws.rs.core.MediaType;

/**
 * @ClassName LuosimaoDuanxinUtil
 * @Description: TODO
 * @Author shenyaqi
 * @Date 2020/9/22
 * @Version V1.0
 **/
@Slf4j
public class LuosimaoDuanxinUtil {

    //短信平台的APIkey
    private static final String DUANXIN_API_KEY = "key-";//需要使用自己的APIkey
    private static final String SPEAK_API_KEY = "key-";//需要使用自己的APIkey
    private static final String SEND_DUANXIN_URL = "http://sms-
api.luosimao.com/v1/send.json";//发送短信的接口
    private static final String SEND_SPEAK_URL = "http://voice-
api.luosimao.com/v1/verify.json";//发送语音的接口
```

```
    private static final String STATUS_URL = "http://sms-
api.luosimao.com/v1/status.json";//查看余额的接口

    public static String SendCode(String phone,String code){


        // just replace key here
        Client client = Client.create();
        client.addFilter(new HTTPBasicAuthFilter(
                "api",DUANXIN_API_KEY));
        WebResource webResource = client.resource(SEND_DUANXIN_URL);
        MultivaluedMapImpl formData = new MultivaluedMapImpl();
        formData.add("mobile", phone);
        formData.add("message", "验证码：" + code + "【铁壳测试】");//注意此处不能修改
        ClientResponse response =
 webResource.type(MediaType.APPLICATION_FORM_URLENCODED).
                post(ClientResponse.class, formData);
        String textEntity = response.getEntity(String.class);
        int status = response.getStatus();
        log.info(textEntity);
        log.info("---------发送短信验证状态------" + status);
        return textEntity;
    }

    public static String sendSpeak(String phone,String code){
        // just replace key here
        Client client = Client.create();
        client.addFilter(new HTTPBasicAuthFilter(
                "api",SPEAK_API_KEY));
        WebResource webResource = client.resource(
                SEND_SPEAK_URL);
        MultivaluedMapImpl formData = new MultivaluedMapImpl();
        formData.add("mobile", phone);
        formData.add("code", code);
        ClientResponse response =
 webResource.type(MediaType.APPLICATION_FORM_URLENCODED).
                post(ClientResponse.class, formData);
        String textEntity = response.getEntity(String.class);
        int status = response.getStatus();
        log.info(textEntity);
        log.info("---------发送语音验证状态------" + status);

        return textEntity;
    }

    private static String getStatus(){
        Client client = Client.create();
        client.addFilter(new HTTPBasicAuthFilter(
                "api",DUANXIN_API_KEY));
        WebResource webResource = client.resource( STATUS_URL );
        MultivaluedMapImpl formData = new MultivaluedMapImpl();
        ClientResponse response =  webResource.get( ClientResponse.class );
        String textEntity = response.getEntity(String.class);
        int status = response.getStatus();

        log.info(textEntity);
        log.info(status + "");
        return textEntity;
        }
    }
```

3.2.4 前端页面



```
methods: {
    createVerifyCode() {// 生成短信验证码
        this.$validator.validate("phone").then(r => {
            if (r) {
                mrshop.http.post("/user-center/user/sendValidCode", {
                    phone:this.user.phone
                }).then(resp => {
                    console.log(resp);
                }).catch(error => console.log(error));
            }
        });
```

3.2.5 UserService

```

    @ApiOperation(value = "给手机号发送验证码")
    @PostMapping(value = "user/sendValidCode")
    Result<JSONObject> sendValidCode(@RequestBody UserDTO userDTO);
```

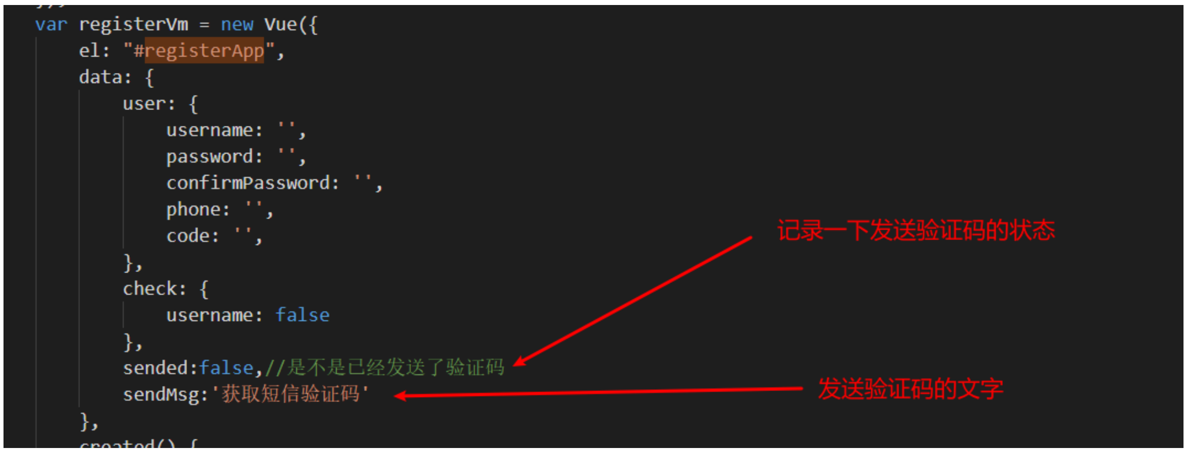
3.2.6 UserServiceImpl

```
    @Override
    public Result<JSONObject> sendValidCode(UserDTO userDTO) {

        //生成随机6位验证码
        String code = (int)((Math.random() * 9 + 1) * 100000) + "";
        //发送短信验证码
        LuosimaoDuanxinUtil.SendCode(userDTO.getPhone(),code);

        return this.setResultSuccess();
    }
```

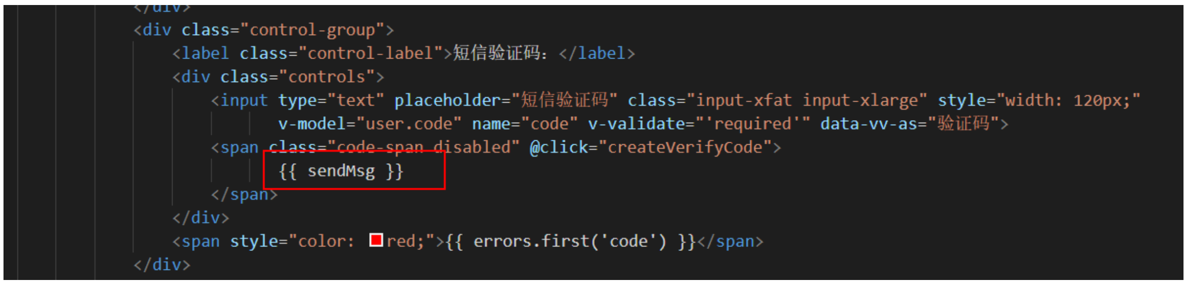
说明:短信验证码只有10条,不够我们测试

```
        //短信条数只有10条,不够我们测试.所以就不发送短信验证码了,直接在控制台打印就可以
        log.debug("向手机号码:{} 发送验证码:{}",userDTO.getPhone(),code);
```

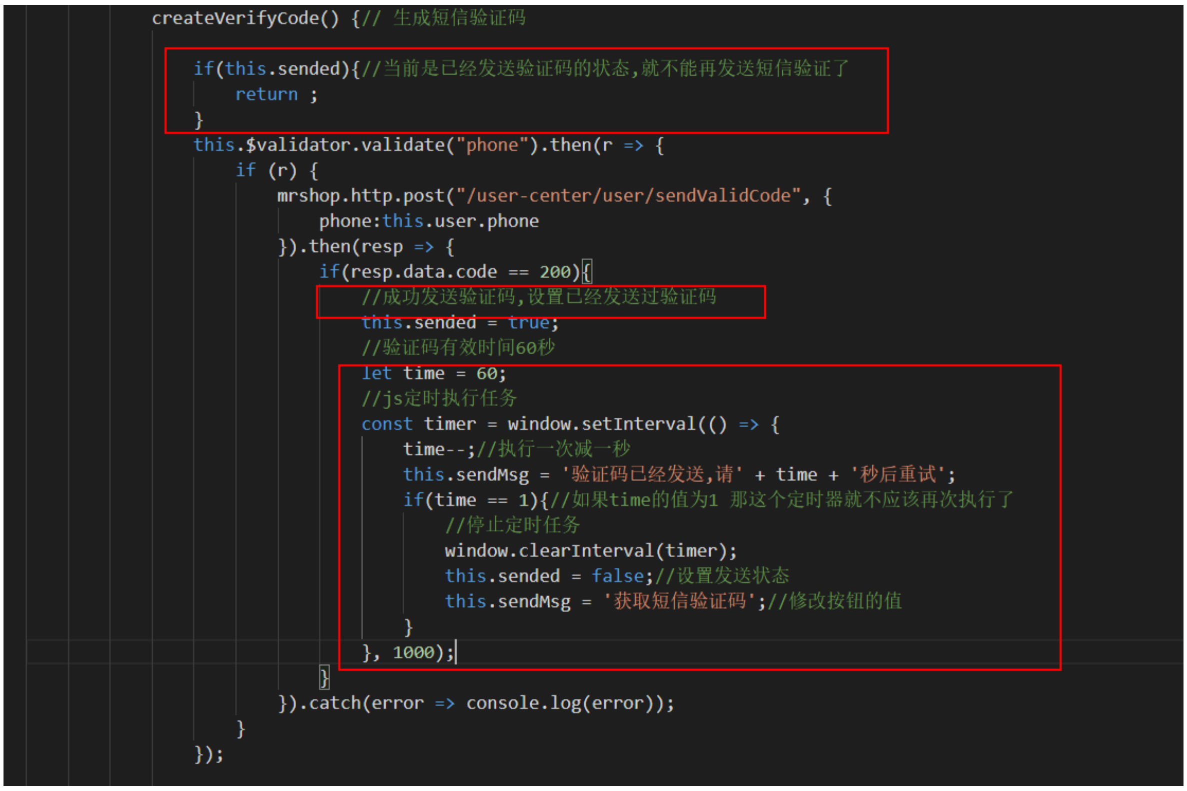
3.2.7 页面效果



```
var registerVm = new Vue({
    el: "#registerApp",
    data: {
        user: {
            username: '',
            password: '',
            confirmPassword: '',
            phone: '',
            code: '',
        },
        check: {
            username: false
        },
        sended:false,//是不是已经发送了验证码
        sendMsg:'获取短信验证码'
    },
```



```
        <div class="control-group">
            <label class="control-label">短信验证码: </label>
            <div class="controls">
                <input type="text" placeholder="短信验证码" class="input-xfat input-xlarge" style="width: 120px;"
                       v-model="user.code" name="code" v-validate="'required'" data-vv-as="验证码">
                <span class="code-span disabled" @click="createVerifyCode">
                    {{ sendMsg }}
                </span>
            </div>
            <span style="color: 🟥red;">{{ errors.first('code') }}</span>
        </div>
```



```
createVerifyCode() {// 生成短信验证码

    if(this.sended){//当前是已经发送验证码的状态,就不能再发送短信验证了
        return ;
    }
    this.$validator.validate("phone").then(r => {
        if (r) {
            mrshop.http.post("/user-center/user/sendValidCode", {
                phone:this.user.phone
            }).then(resp => {
                if(resp.data.code == 200){
                    //成功发送验证码,设置已经发送过验证码
                    this.sended = true;
                    //验证码有效时间60秒
                    let time = 60;
                    //js定时执行任务
                    const timer = window.setInterval(() => {
                        time--;//执行一次减一秒
                        this.sendMsg = '验证码已经发送,请' + time + '秒后重试';
                        if(time == 1){//如果time的值为1 那这个定时器就不应该再次执行了
                            //停止定时任务
                            window.clearInterval(timer);
                            this.sended = false;//设置发送状态
                            this.sendMsg = '获取短信验证码';//修改按钮的值
                        }
                    }, 1000);
                }
            }).catch(error => console.log(error));
        }
    });
```

```
        createVerifyCode() {// 生成短信验证码

            if(this.sended){//当前是已经发送验证码的状态,就不能再发送短信验证了
                return ;
            }
            this.$validator.validate("phone").then(r => {
                if (r) {
                    mrshop.http.post("/user-center/user/sendValidCode", {
                        phone:this.user.phone
                    }).then(resp => {
                        if(resp.data.code == 200){
                            //成功发送验证码,设置已经发送过验证码
                            this.sended = true;
```

```
                                //验证码有效时间60秒
                                let time = 60;
                                //js定时执行任务
                                const timer = window.setInterval(() => {
                                    time--;//执行一次减一秒
                                    this.sendMsg = '验证码已经发送,请' + time + '秒后重试';

                                    if(time == 1){//如果time的值为1 那这个定时器就不应该再次执行了
                                        //停止定时任务
                                        window.clearInterval(timer);
                                        this.sended = false;//设置发送状态
                                        this.sendMsg = '获取短信验证码';//修改按钮的值
                                    }
                                }, 1000);
                            }
                        }).catch(error => console.log(error));
                    }
                });

            },
```

3.3 校验手机验证码

3.3.1 redis

3.3.1.1 redis介绍

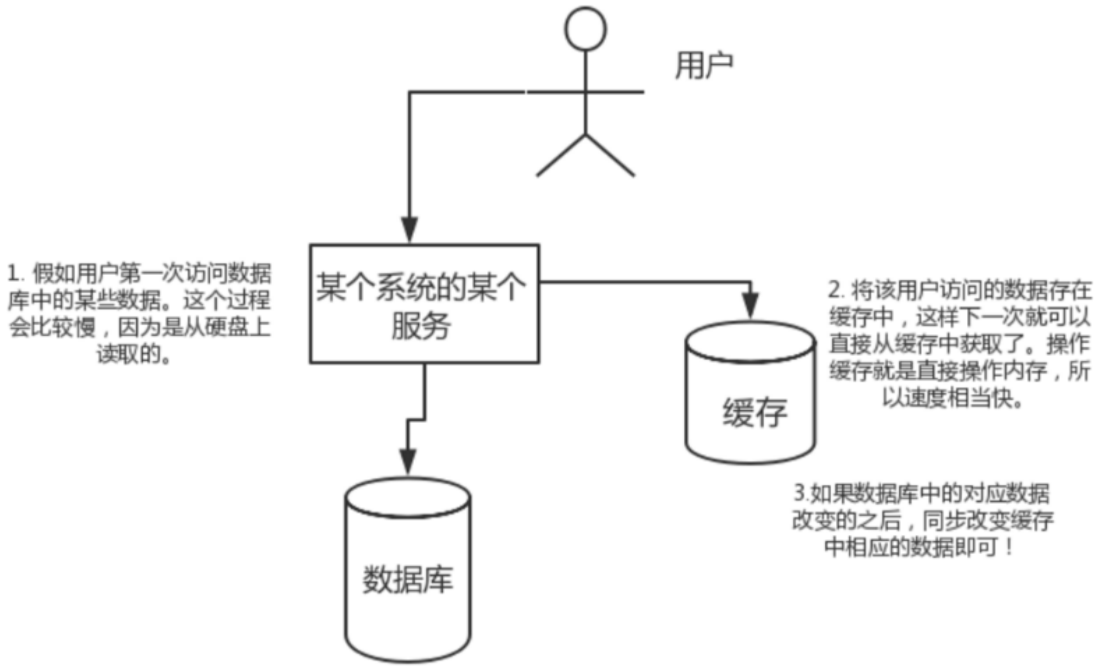
简单来说 redis 就是一个数据库，不过与传统数据库不同的是 redis 的数据是存在内存中的，所以读写速度非常快，因此 redis 被广泛应用于缓存方向。另外，redis 也经常用来做分布式锁。redis 提供了多种数据类型来支持不同的业务场景。除此之外，redis 支持事务 、持久化、LUA脚本、LRU驱动事件、多种集群方案。

3.3.1.2 为什么要用 redis？为什么要用缓存?

主要从“高性能”和“高并发”这两点来看待这个问题。

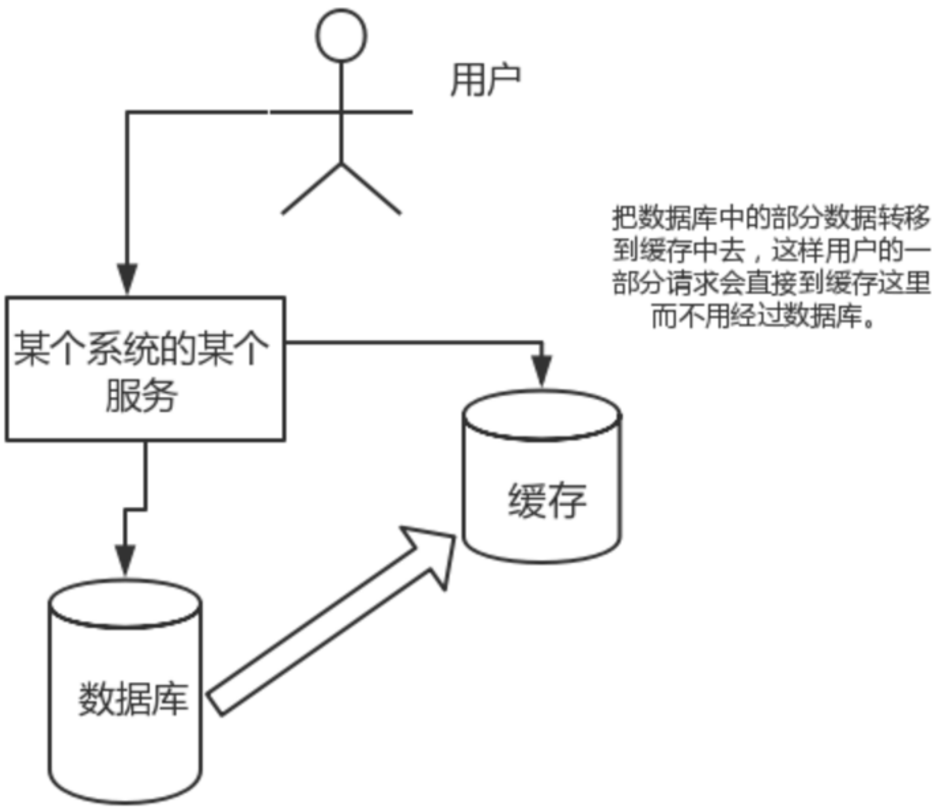
> 高性能：

假如用户第一次访问数据库中的某些数据。这个过程会比较慢，因为是从硬盘上读取的。将该用户访问的数据存在缓存中，这样下一次再访问这些数据的时候就可以直接从缓存中获取了。操作缓存就是直接操作内存，所以速度相当快。如果数据库中的对应数据改变的之后，同步改变缓存中相应的数据即可


用户
1. 假如用户第一次访问数据库中的某些数据。这个过程会比较慢，因为是从硬盘上读取的。
某个系统的某个服务
2. 将该用户访问的数据存在缓存中，这样下一次就可以直接从缓存中获取了。操作缓存就是直接操作内存，所以速度相当快。
缓存
数据库
3.如果数据库中的对应数据改变的之后，同步改变缓存中相应的数据即可！

> 高并发：

直接操作缓存能够承受的请求是远远大于直接访问数据库的，所以我们可以考虑把数据库中的部分数据转移到缓存中去，这样用户的一部分请求会直接到缓存这里而不用经过数据库。


用户
把数据库中的部分数据转移到缓存中去，这样用户的一部分请求会直接到缓存这里而不用经过数据库。
某个系统的某个服务
缓存
数据库

我们现在使用redis存储一下平台发送到指定手机号上的验证码

3.3.2.3 redis安装

linux下新建redis目录

下载linux压缩包

```
wget http://download.redis.io/releases/redis-5.0.5.tar.gz
```

解压压缩包

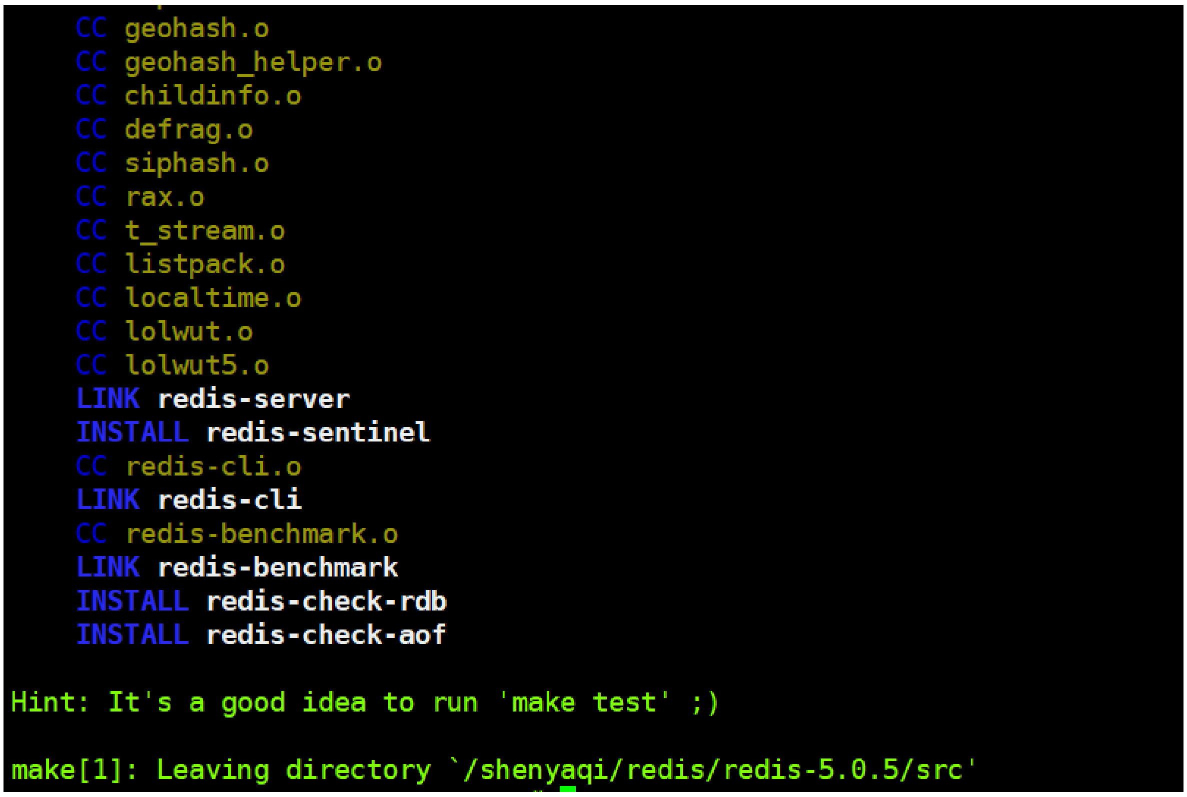
```
tar -zxvf redis-5.0.5.tar.gz
```

进入刚刚解压的目录

```
cd redis-5.0.5/
```

编译

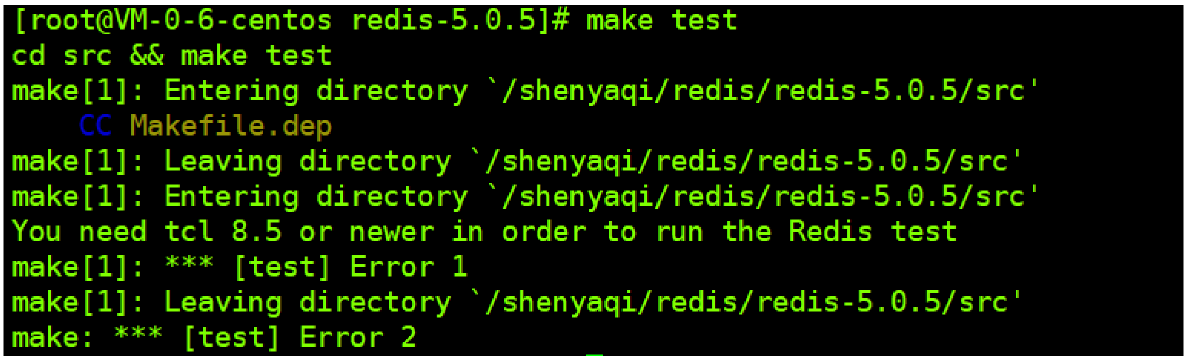
```
make
```



```
    CC geohash.o
    CC geohash_helper.o
    CC childinfo.o
    CC defrag.o
    CC siphash.o
    CC rax.o
    CC t_stream.o
    CC listpack.o
    CC localtime.o
    CC lolwut.o
    CC lolwut5.o
    LINK redis-server
    INSTALL redis-sentinel
    CC redis-cli.o
    LINK redis-cli
    CC redis-benchmark.o
    LINK redis-benchmark
    INSTALL redis-check-rdb
    INSTALL redis-check-aof

Hint: It's a good idea to run 'make test' ;)

make[1]: Leaving directory `/shenyaqi/redis/redis-5.0.5/src'
```

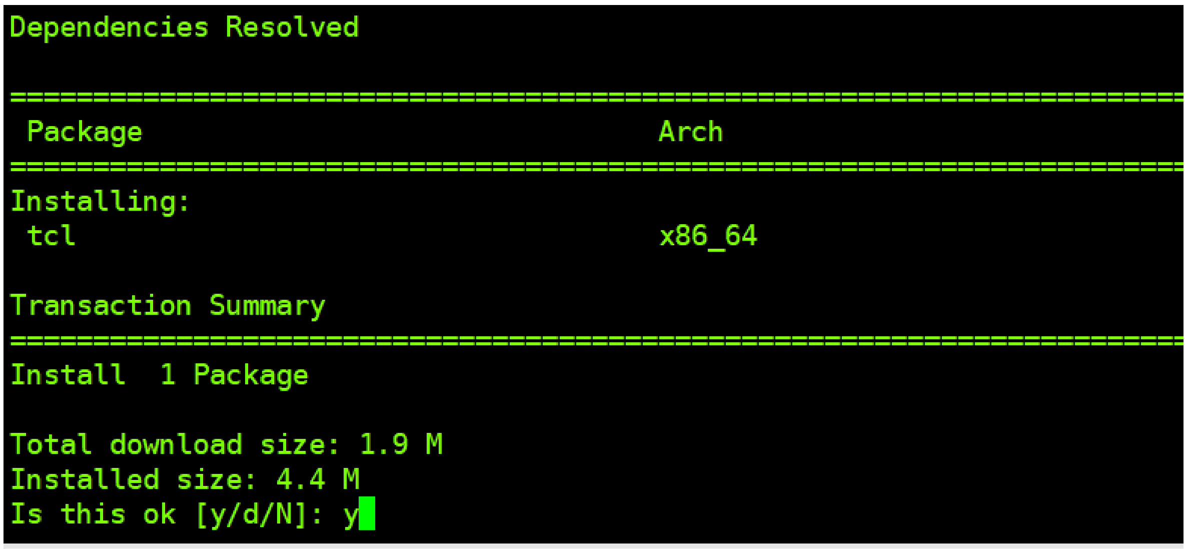
测试编译

```
make test
```

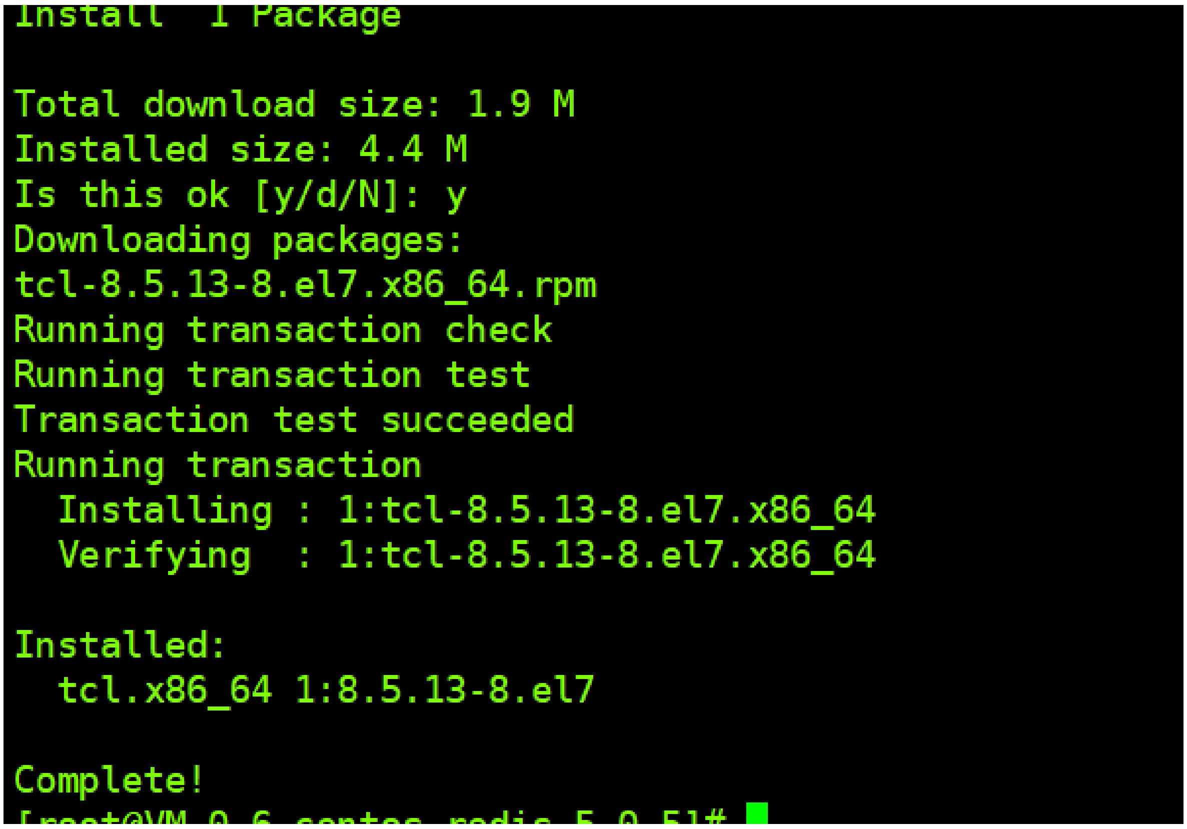


```
[root@VM-0-6-centos redis-5.0.5]# make test
cd src && make test
make[1]: Entering directory `/shenyaqi/redis/redis-5.0.5/src'
    CC Makefile.dep
make[1]: Leaving directory `/shenyaqi/redis/redis-5.0.5/src'
make[1]: Entering directory `/shenyaqi/redis/redis-5.0.5/src'
You need tcl 8.5 or newer in order to run the Redis test
make[1]: *** [test] Error 1
make[1]: Leaving directory `/shenyaqi/redis/redis-5.0.5/src'
make: *** [test] Error 2
```

```
yum install tcl
```



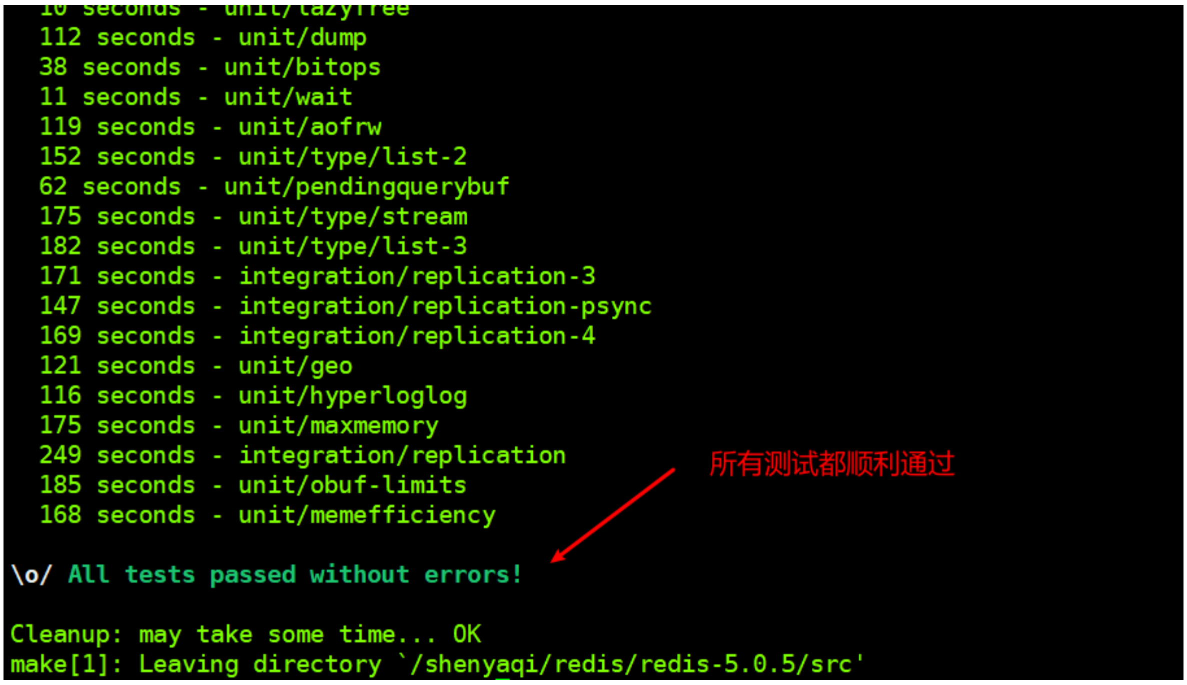
```
Dependencies Resolved

================================================================================
 Package                                      Arch
================================================================================
Installing:
 tcl                                          x86_64

Transaction Summary
================================================================================
Install  1 Package

Total download size: 1.9 M
Installed size: 4.4 M
Is this ok [y/d/N]: y
```



```
Install  1 Package

Total download size: 1.9 M
Installed size: 4.4 M
Is this ok [y/d/N]: y
Downloading packages:
tcl-8.5.13-8.el7.x86_64.rpm
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : 1:tcl-8.5.13-8.el7.x86_64
  Verifying  : 1:tcl-8.5.13-8.el7.x86_64

Installed:
  tcl.x86_64 1:8.5.13-8.el7

Complete!
[root@VM_0_6_centos redis-5.0.5]# 
```

再次测试编译

```
make test
```



```
  10 seconds - unit/lazyfree
  112 seconds - unit/dump
  38 seconds - unit/bitops
  11 seconds - unit/wait
  119 seconds - unit/aofrw
  152 seconds - unit/type/list-2
  62 seconds - unit/pendingquerybuf
  175 seconds - unit/type/stream
  182 seconds - unit/type/list-3
  171 seconds - integration/replication-3
  147 seconds - integration/replication-psync
  169 seconds - integration/replication-4
  121 seconds - unit/geo
  116 seconds - unit/hyperloglog
  175 seconds - unit/maxmemory
  249 seconds - integration/replication
  185 seconds - unit/obuf-limits
  168 seconds - unit/memefficiency

\o/ All tests passed without errors!

Cleanup: may take some time... OK
make[1]: Leaving directory `/shenyaqi/redis/redis-5.0.5/src'
```

创建etc和bin文件夹

```
mkdir etc
mkdir bin
```

将redis配置文件移动到etc文件夹下

```
mv redis.conf etc/
```

进入src文件夹

```
cd src/
```

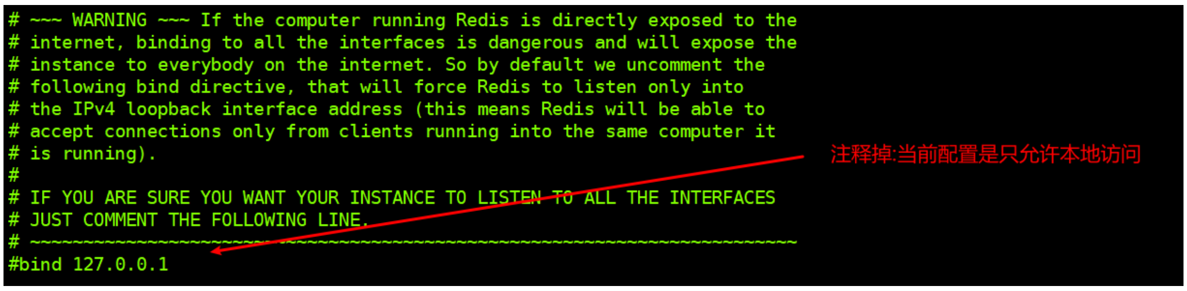
将我们有可能使用到的脚本放到bin文件夹

```
mv mkreleasehdr.sh redis-benchmark redis-check-aof redis-check-rdb redis-cli
redis-server ../bin/
```

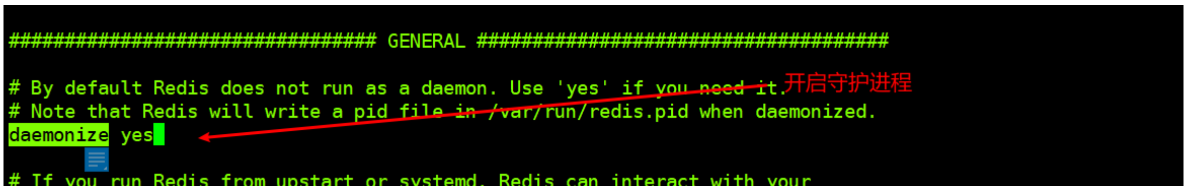
进入etc文件夹

修改redis.conf

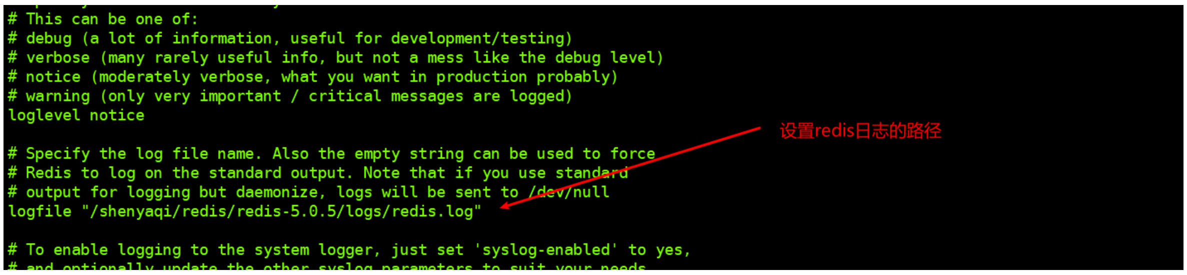
```
vi redis.conf
```



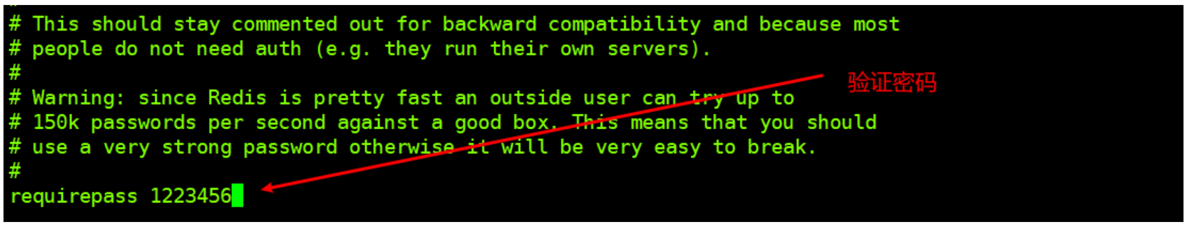
```
# ~~~ WARNING ~~~ If the computer running Redis is directly exposed to the
# internet, binding to all the interfaces is dangerous and will expose the
# instance to everybody on the internet. So by default we uncomment the
# following bind directive, that will force Redis to listen only into
# the IPv4 loopback interface address (this means Redis will be able to
# accept connections only from clients running into the same computer it
# is running).
#
# IF YOU ARE SURE YOU WANT YOUR INSTANCE TO LISTEN TO ALL THE INTERFACES
# JUST COMMENT THE FOLLOWING LINE.
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#bind 127.0.0.1
```



```
################################ GENERAL #####################################

# By default Redis does not run as a daemon. Use 'yes' if you need it.
# Note that Redis will write a pid file in /var/run/redis.pid when daemonized.
daemonize yes

# If you run Redis from upstart or systemd, Redis can interact with your
```



```
# This can be one of:
# debug (a lot of information, useful for development/testing)
# verbose (many rarely useful info, but not a mess like the debug level)
# notice (moderately verbose, what you want in production probably)
# warning (only very important / critical messages are logged)
loglevel notice

# Specify the log file name. Also the empty string can be used to force
# Redis to log on the standard output. Note that if you use standard
# output for logging but daemonize, logs will be sent to /dev/null
logfile "/shenyaqi/redis/redis-5.0.5/logs/redis.log"

# To enable logging to the system logger, just set 'syslog-enabled' to yes,
# and optionally update the other syslog parameters to suit your needs.
```



```
# This should stay commented out for backward compatibility and because most
# people do not need auth (e.g. they run their own servers).
#
# Warning: since Redis is pretty fast an outside user can try up to
# 150k passwords per second against a good box. This means that you should
# use a very strong password otherwise it will be very easy to break.
#
requirepass 1223456
```

在redis_home下创建logs文件夹

```
mkdir logs
```

文件夹内创建redis.log文件

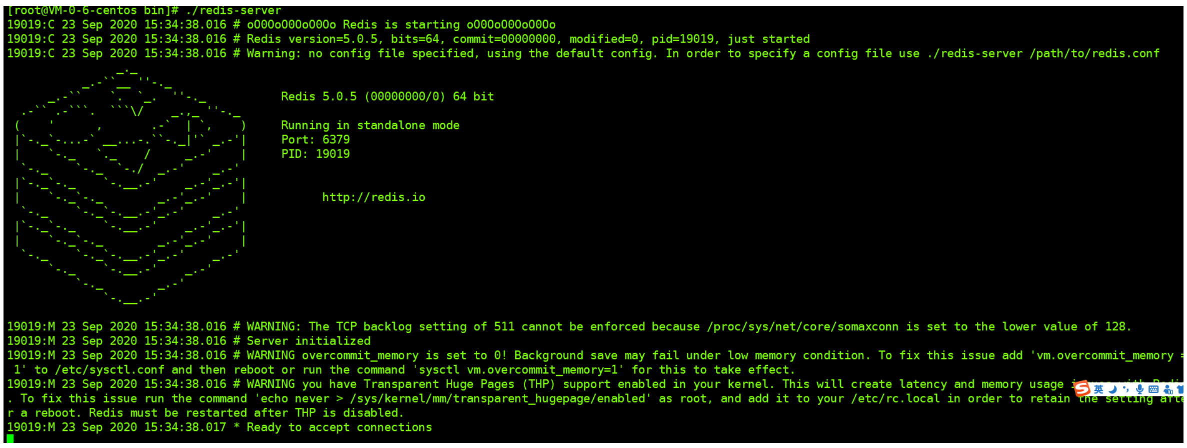
```
touch redis.log
```

进入bin文件夹

启动redis服务

```
./redis-server
./redis-server ../etc/redis.conf #后台运行
ps -ef | grep redis #查看redis进程
```

进入logs文件夹查看日志



```
[root@VM-0-6-centos bin]# ./redis-server
19019:C 23 Sep 2020 15:34:38.016 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
19019:C 23 Sep 2020 15:34:38.016 # Redis version=5.0.5, bits=64, commit=00000000, modified=0, pid=19019, just started
19019:C 23 Sep 2020 15:34:38.016 # Warning: no config file specified, using the default config. In order to specify a config file use ./redis-server /path/to/redis.conf

                                    Redis 5.0.5 (00000000/0) 64 bit

                                    Running in standalone mode
                                    Port: 6379
                                    PID: 19019


                                          http://redis.io


19019:M 23 Sep 2020 15:34:38.016 # WARNING: The TCP backlog setting of 511 cannot be enforced because /proc/sys/net/core/somaxconn is set to the lower value of 128.
19019:M 23 Sep 2020 15:34:38.016 # Server initialized
19019:M 23 Sep 2020 15:34:38.016 # WARNING overcommit_memory is set to 0! Background save may fail under low memory condition. To fix this issue add 'vm.overcommit_memory =
 1' to /etc/sysctl.conf and then reboot or run the command 'sysctl vm.overcommit_memory=1' for this to take effect.
19019:M 23 Sep 2020 15:34:38.016 # WARNING you have Transparent Huge Pages (THP) support enabled in your kernel. This will create latency and memory usage i
. To fix this issue run the command 'echo never > /sys/kernel/mm/transparent_hugepage/enabled' as root, and add it to your /etc/rc.local in order to retain the setting afte
r a reboot. Redis must be restarted after THP is disabled.
19019:M 23 Sep 2020 15:34:38.017 * Ready to accept connections
```

进入bin目录

执行客户端脚本

```
./redis-cli
auth 123456 #验证密码
```

3.3.2 项目整合redis

3.3.2.1 api-user

3.3.2.1.1 pom.xml

```
    <dependencies>
        <!--springboot整合redis-->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-redis</artifactId>
        </dependency>
        <!--redis需要的对象池 redis可以将存储对象-->
        <dependency>
            <groupId>org.apache.commons</groupId>
            <artifactId>commons-pool2</artifactId>
            <version>2.8.0</version>
        </dependency>
    </dependencies>
```

3.3.2.1.2 config包下新建RedisConfig

```
import com.fasterxml.jackson.annotation.JsonAutoDetect;
import com.fasterxml.jackson.annotation.JsonTypeInfo;
import com.fasterxml.jackson.annotation.PropertyAccessor;
import com.fasterxml.jackson.databind.ObjectMapper;
import
com.fasterxml.jackson.databind.jsontype.impl.LaissezFaireSubTypeValidator;
import org.springframework.cache.CacheManager;
import org.springframework.cache.annotation.CachingConfigurerSupport;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.Primary;
import org.springframework.data.redis.cache.RedisCacheConfiguration;
import org.springframework.data.redis.cache.RedisCacheManager;
import org.springframework.data.redis.connection.RedisConnectionFactory;
import org.springframework.data.redis.core.RedisTemplate;
import org.springframework.data.redis.serializer.Jackson2JsonRedisSerializer;
import org.springframework.data.redis.serializer.RedisSerializationContext;
import org.springframework.data.redis.serializer.RedisSerializer;
import org.springframework.data.redis.serializer.StringRedisSerializer;

import java.nio.charset.Charset;
import java.time.Duration;

/**
 * @ClassName RedisConfig
 * @Description: TODO
 * @Author shenyaqi
 * @Date 2020/9/24
 * @Version V1.0
 **/
@Configuration
```

```
public class RedisConfig extends CachingConfigurerSupport {
    @Bean
    @Primary
    public RedisTemplate<String, Object> redisTemplate(RedisConnectionFactory
factory) {

        RedisTemplate<String, Object> template = new RedisTemplate<>();

        RedisSerializer<String> redisSerializer = new
StringRedisSerializer(Charset.forName("UTF8"));

        Jackson2JsonRedisSerializer jackson2JsonRedisSerializer = new
Jackson2JsonRedisSerializer(Object.class);
        ObjectMapper om = new ObjectMapper();
        om.setVisibility(PropertyAccessor.ALL, JsonAutoDetect.Visibility.ANY);
        //om.enableDefaultTyping(ObjectMapper.DefaultTyping.NON_FINAL);//方法已经
过时,我印象当中这个方法是有漏洞的

 om.activateDefaultTyping(LaissezFaireSubTypeValidator.instance,ObjectMapper.Def
aultTyping.NON_FINAL, JsonTypeInfo.As.WRAPPER_ARRAY);
        jackson2JsonRedisSerializer.setObjectMapper(om);

        template.setConnectionFactory(factory);
        //key序列化方式
        template.setKeySerializer(redisSerializer);
        //value序列化
        template.setValueSerializer(jackson2JsonRedisSerializer);
        //value hashmap序列化
        template.setHashValueSerializer(jackson2JsonRedisSerializer);

        return template;
    }

    @Bean
    public CacheManager cacheManager(RedisConnectionFactory factory) {
        RedisSerializer<String> redisSerializer = new StringRedisSerializer();
        Jackson2JsonRedisSerializer jackson2JsonRedisSerializer = new
Jackson2JsonRedisSerializer(Object.class);

        //解决查询缓存转换异常的问题
        ObjectMapper om = new ObjectMapper();
        om.setVisibility(PropertyAccessor.ALL, JsonAutoDetect.Visibility.ANY);
        //om.enableDefaultTyping(ObjectMapper.DefaultTyping.NON_FINAL);

 om.activateDefaultTyping(LaissezFaireSubTypeValidator.instance,ObjectMapper.Def
aultTyping.NON_FINAL, JsonTypeInfo.As.WRAPPER_ARRAY);
        jackson2JsonRedisSerializer.setObjectMapper(om);

        // 配置序列化（解决乱码的问题）,过期时间30秒
        RedisCacheConfiguration config =
RedisCacheConfiguration.defaultCacheConfig()
                .entryTtl(Duration.ofSeconds(30))

.serializeKeysWith(RedisSerializationContext.SerializationPair.fromSerializer(re
disSerializer))

.serializeValuesWith(RedisSerializationContext.SerializationPair.fromSerializer(
jackson2JsonRedisSerializer))
```

```
                .disableCachingNullValues();

        RedisCacheManager cacheManager = RedisCacheManager.builder(factory)
                .cacheDefaults(config)
                .build();
        return cacheManager;
    }
}
```

3.3.2.2 service-user

application.yml

```yaml
spring:
  # 整合redis配置
  redis:
    # 数据库标识，可以配置多个redis使用不同的标识区分
    database: 0
    # redisIP地址
    host: 127.0.0.1
    # redis端口号
    port: 6379
    # redis密码
    password: 123456
    # redis连接池的配置
    jedis:
      pool:
        #最大连接数据库连接数,设 0 为没有限制
        max-active: 8
        #最大等待连接中的数量,设 0 为没有限制
        max-idle: 8
        #最大建立连接等待时间。如果超过此时间将接到异常。设为-1表示无限制。
        max-wait: -1ms
        #最小等待连接中的数量,设 0 为没有限制
        min-idle: 0
```

3.3.2.2.1 com.baidu.shop包下新建redis.repository包

3.3.2.2.2 包下新建RedisRepository

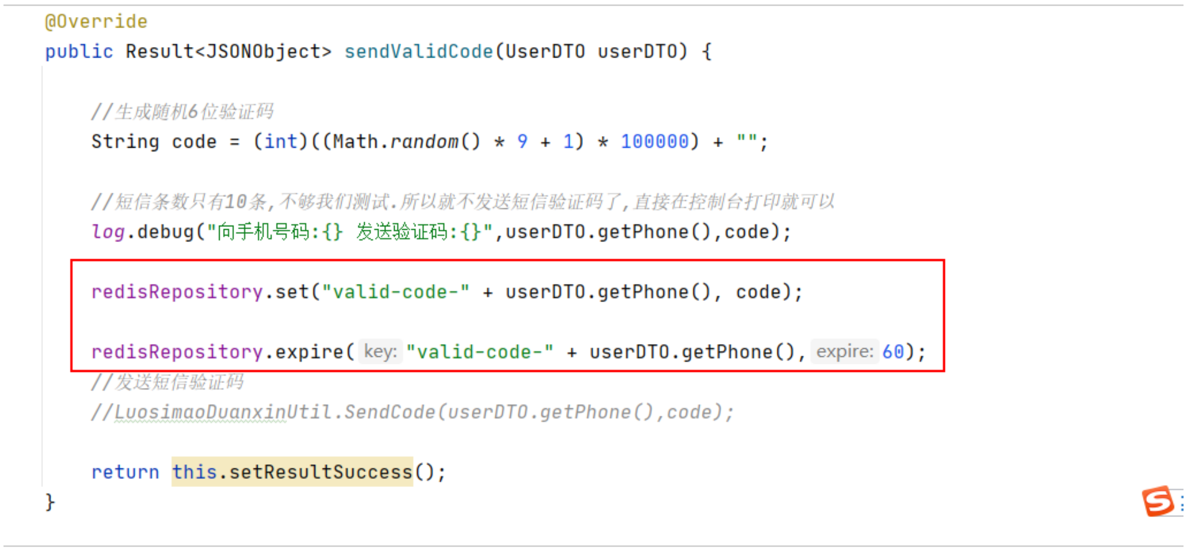
```java
import com.baidu.shop.utils.JSONUtil;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.dao.DataAccessException;
import org.springframework.data.redis.connection.RedisConnection;
import org.springframework.data.redis.core.RedisCallback;
import org.springframework.data.redis.core.StringRedisTemplate;
import org.springframework.data.redis.serializer.RedisSerializer;
import org.springframework.stereotype.Component;

import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.concurrent.TimeUnit;

/**
```

```
 * @ClassName RedisRepository
 * @Description: TODO
 * @Author shenyaqi
 * @Date 2020/9/24
 * @Version V1.0
 **/
@Component
public class RedisRepository {

    //--注意：此处不能使用Resource注解，因为在RedisConfig line:30行中使用@Bean注解，方法
的返回值是RedisTemplate @Resource默认按名称自动注入，会与我们定义的redisTemplate冲突
    @Autowired
    private StringRedisTemplate redisTemplate;

    /**
     * 获取序列化工具
     * @return
     */
    private RedisSerializer<String> getSerializer(){

        return redisTemplate.getStringSerializer();
    }

    /**
     * 放入string类型的值
     * @param key
     * @param value
     * @return
     */
    public boolean set(final String key, final String value) {
        return redisTemplate.execute(new RedisCallback<Boolean>() {

            @Override
            public Boolean doInRedis(RedisConnection connection) throws
DataAccessException {

                return connection.set(getSerializer().serialize(key),
getSerializer().serialize(value));
            }
        });
    }

    /**
     * 放入对象类型的值
     * @param key
     * @param obj
     * @return
     */
    public boolean setObj(final String key, final Object obj) {

        return redisTemplate.execute(new RedisCallback<Boolean>() {

            @Override
            public Boolean doInRedis(RedisConnection connection) throws
DataAccessException {

                return connection.set(getSerializer().serialize(key),
getSerializer().serialize(JSONUtil.toJsonString(obj)));
```

```
            }
        });
    }

    /**
     * 获取string类型的值
     * @param key
     * @return
     */
    public String get(final String key) {

        return redisTemplate.execute(new RedisCallback<String>() {
            @Override
            public String doInRedis(RedisConnection connection) throws
DataAccessException {

                byte[] value = connection.get(getSerializer().serialize(key));

                return getSerializer().deserialize(value);
            }
        });
    }

    /**
     * 获取对象类型的值
     * @param key
     * @param clazz
     * @param <T>
     * @return
     */
    public <T> T getObj(final String key,Class<T> clazz) {

        String result = redisTemplate.execute(new RedisCallback<String>() {

            @Override
            public String doInRedis(RedisConnection connection) throws
DataAccessException {

                byte[] value = connection.get(getSerializer().serialize(key));

                return getSerializer().deserialize(value);
            }
        });

        Object o = JSONUtil.toBean(result, clazz);

        if(clazz.isInstance(o)){

            return clazz.cast(o);
        }

        return null;
    }

    /**
     * 给key值设置过期时间
     * @param key
     * @param expire
     * @return
     */
    public boolean expire(final String key, long expire) {
        return redisTemplate.expire(key, expire, TimeUnit.SECONDS);
    }

    /**
     * 放入list类型的值
     * @param key
     * @param list
     * @param <T>
     * @return
     */
    public <T> boolean setList(String key, List<T> list) {

        return set(key, JSONUtil.toJsonString(list));
    }

    /**
     * 获取list类型的值
     * @param key
     * @param clz
     * @param <T>
     * @return
     */
    public <T> List<T> getList(String key, Class<T> clz) {

        String json = get(key);

        if (json != null) {

            List<T> list = JSONUtil.toList(json, clz);
            return list;
        }
        return null;
    }

    /**
     * 操作队列
     * @param key
     * @param obj
     * @return
     */
    public long rpush(final String key, Object obj) {
        final String value = JSONUtil.toJsonString(obj);
        long result = redisTemplate.execute(new RedisCallback<Long>() {
            @Override
            public Long doInRedis(RedisConnection connection) throws
DataAccessException {

                long count = connection.rPush(getSerializer().serialize(key),
getSerializer().serialize(value));
                return count;
            }
        });
        return result;
    }
    /**
     * 删除队列
     * @param key
     * @return
     */
    public String lpop(final String key) {
        String result = redisTemplate.execute(new RedisCallback<String>() {
            @Override
            public String doInRedis(RedisConnection connection) throws
DataAccessException {

                byte[] res = connection.lPop(getSerializer().serialize(key));
                return getSerializer().deserialize(res);
            }
        });
        return result;
    }

    /**
     * 通过key值删除缓存数据
     * @param key
     * @return
     */
    public boolean del(String key) {
        // TODO Auto-generated method stub
        return redisTemplate.delete(key);
    }

    /**
     * 存入hash类型的值
     * @param key
     * @param mapKey
     * @param value
     * @return
     */
    public boolean setHash(final String key,final String mapKey, final String
value) {

        return redisTemplate.execute(new RedisCallback<Boolean>() {
            @Override
            public Boolean doInRedis(RedisConnection connection) throws
DataAccessException {
                // TODO Auto-generated method stub

                return connection.hSet(getSerializer().serialize(key),
getSerializer().serialize(mapKey), getSerializer().serialize(value));
            }
        });
    }

    /**
     * 通过redis的key和hashkey删除value
     * @param key
     * @param mapKey
     * @return
     */
    public boolean delHash(final String key,final String mapKey) {
        return redisTemplate.execute(new RedisCallback<Boolean>() {
            @Override
            public Boolean doInRedis(RedisConnection connection) throws
DataAccessException {
                // TODO Auto-generated method stub
                Long aLong = connection.hDel(getSerializer().serialize(key),
getSerializer().serialize(mapKey));
                return aLong != 0;
            }
        });
    }

    /**
     * 通过rediskey删除hash
     * @param key
     * @return
     */
    public boolean delHash(final String key) {

        return redisTemplate.execute(new RedisCallback<Boolean>() {
            @Override
            public Boolean doInRedis(RedisConnection connection) throws
DataAccessException {
                // TODO Auto-generated method stub
                Long aLong = connection.hDel(getSerializer().serialize(key));
                return aLong != 0;
            }
        });
    }


    /**
     * 根据redis的key和hask的key获取hash对应的值
     * @param key
     * @param mapKey
     * @param clazz
     * @param <T>
     * @return
     */
    public <T> T getHash(final String key,final String mapKey,Class<T> clazz) {

        String result = redisTemplate.execute(new RedisCallback<String>() {
            @Override
            public String doInRedis(RedisConnection connection) throws
DataAccessException {

                byte[] bytes = connection.hGet(getSerializer().serialize(key),
getSerializer().serialize(mapKey));

                return getSerializer().deserialize(bytes);
            }
        });

        Object o = JSONUtil.toBean(result, clazz);
        if(clazz.isInstance(o)){
            return clazz.cast(o);
        }
        return null;
    }

    /**
     * 根据redis 的key值获取hash的entry
     * @param key
     * @return
     */
    public Map<String, String> getHash(final String key) {

        Map<byte[], byte[]> result = (Map<byte[], byte[]>)
redisTemplate.execute(new RedisCallback<Map<byte[], byte[]>>() {
            @Override
            public Map<byte[], byte[]> doInRedis(RedisConnection connection)
throws DataAccessException {

                Map<byte[], byte[]> map =
connection.hGetAll(getSerializer().serialize(key));
                return map;
            }
        });

        Map<String, String> map = new HashMap<String, String>();
        for (Map.Entry<byte[], byte[]> entry : result.entrySet()){


 map.put(getSerializer().deserialize(entry.getKey()),getSerializer().deserialize
(entry.getValue()));
        }

        return map;
    }
}
```

3.3.2.2.3 UserServiceImpl

在发送手机验证码同时将验证码存到redis库中,并设置过期时间为60秒

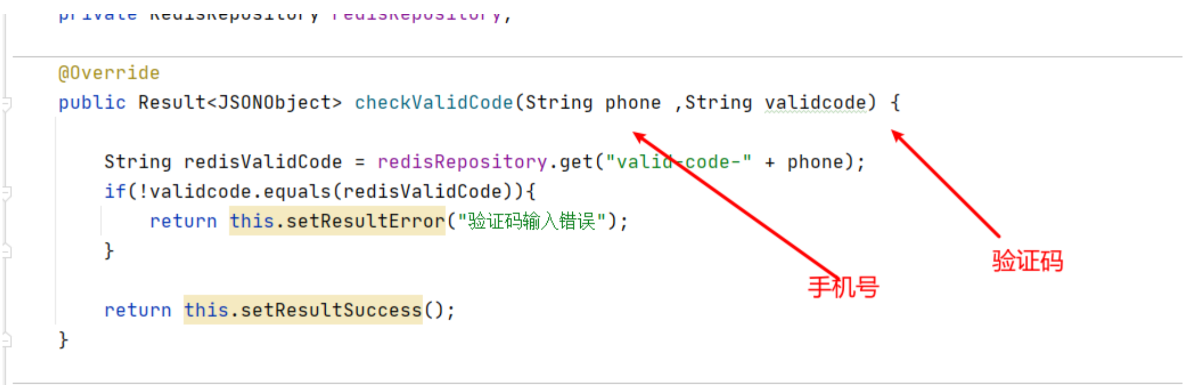


```
@Override
public Result<JSONObject> sendValidCode(UserDTO userDTO) {

    //生成随机6位验证码
    String code = (int)((Math.random() * 9 + 1) * 100000) + "";

    //短信条数只有10条,不够我们测试.所以就不发送短信验证码了,直接在控制台打印就可以
    log.debug("向手机号码:{} 发送验证码:{}",userDTO.getPhone(),code);

    redisRepository.set("valid-code-" + userDTO.getPhone(), code);

    redisRepository.expire( key: "valid-code-" + userDTO.getPhone(), expire: 60);
    //发送短信验证码
    //LuosimaoDuanxinUtil.SendCode(userDTO.getPhone(),code);

    return this.setResultSuccess();
}
```

```
    @Autowired
    private RedisRepository redisRepository;

    @Override
    public Result<JSONObject> sendValidCode(UserDTO userDTO) {

        //生成随机6位验证码
        String code = (int)((Math.random() * 9 + 1) * 100000) + "";

        //短信条数只有10条,不够我们测试.所以就不发送短信验证码了,直接在控制台打印就可以
        log.debug("向手机号码:{} 发送验证码:{}",userDTO.getPhone(),code);

        redisRepository.set("valid-code-" + userDTO.getPhone(), code);

        redisRepository.expire("valid-code-" + userDTO.getPhone(),60);
        //发送短信验证码
        //LuosimaoDuanxinUtil.SendCode(userDTO.getPhone(),code);

        return this.setResultSuccess();
    }
```

3.2.2.2.4 提供一个校验验证码的接口

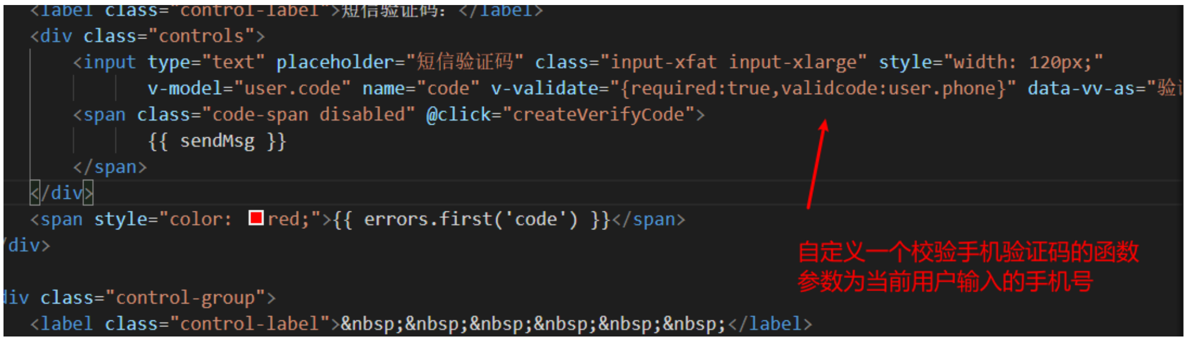
接口的方法就没有在文档里面写了



```
    @Override
    public Result<JSONObject> checkValidCode(String phone ,String validcode) {

        String redisValidCode = redisRepository.get("valid-code-" + phone);
        if(!validcode.equals(redisValidCode)){
            return this.setResultError("验证码输入错误");
        }

        return this.setResultSuccess();
    }
```

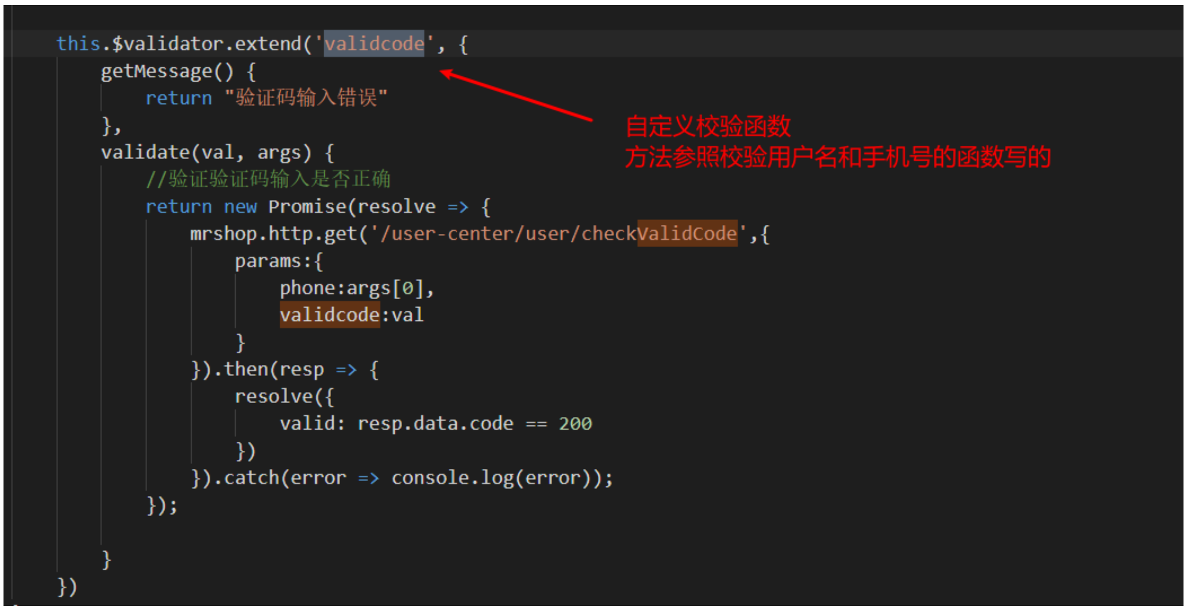
```
    @Override
    public Result<JSONObject> checkValidCode(String phone ,String validcode) {

        String redisValidCode = redisRepository.get("valid-code-" + phone);
        if(!validcode.equals(redisValidCode)){
            return this.setResultError("验证码输入错误");
        }

        return this.setResultSuccess();
    }
```

3.3.2.3 页面校验



```
  <label class="control-label">短信验证码: </label>
  <div class="controls">
      <input type="text" placeholder="短信验证码" class="input-xfat input-xlarge" style="width: 120px;"
             v-model="user.code" name="code" v-validate="{required:true,validcode:user.phone}" data-vv-as="验
      <span class="code-span disabled" @click="createVerifyCode">
             {{ sendMsg }}
      </span>
  </div>
  <span style="color: 🟥red;">{{ errors.first('code') }}</span>
/div>

iv class="control-group">
  <label class="control-label">&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;</label>
```

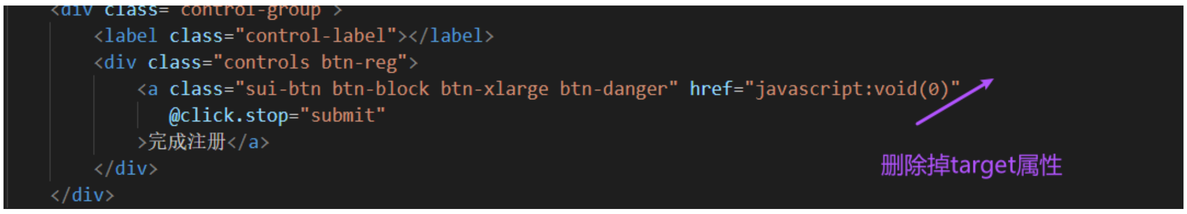
```
<input type="text" placeholder="短信验证码" class="input-xfat input-xlarge"
style="width: 120px;"
                                   v-model="user.code" name="code" v-validate="
{required:true,validcode:user.phone}" data-vv-as="验证码">
```



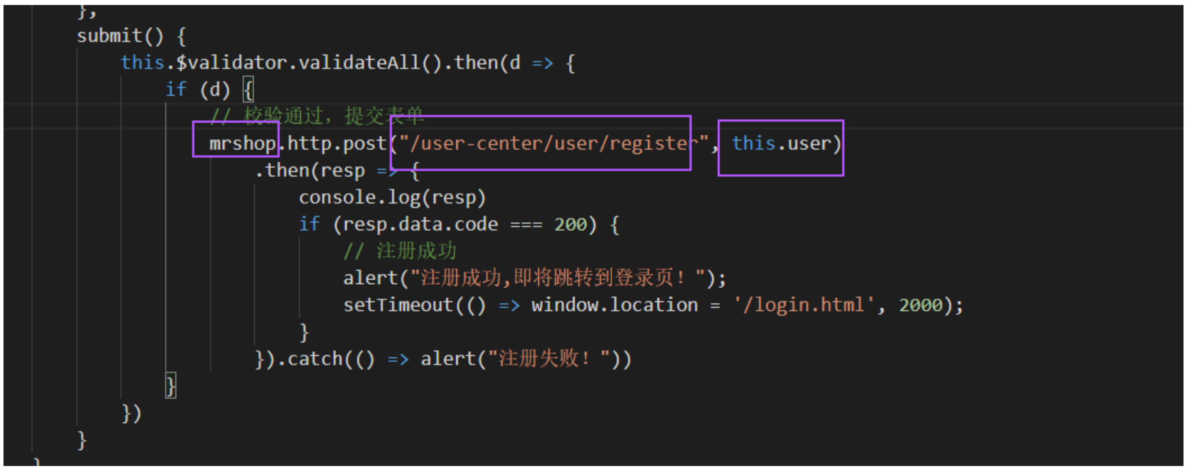
```
this.$validator.extend('validcode', {
    getMessage() {
        return "验证码输入错误"
    },
    validate(val, args) {
        //验证验证码输入是否正确
        return new Promise(resolve => {
            mrshop.http.get('/user-center/user/checkValidCode',{
                params:{
                    phone:args[0],
                    validcode:val
                }
            }).then(resp => {
                resolve({
                    valid: resp.data.code == 200
                })
            }).catch(error => console.log(error));
        });

    }
})
```

```
            this.$validator.extend('validcode', {
                getMessage() {
                    return "验证码输入错误"
                },
                validate(val, args) {
                    //验证验证码输入是否正确
                    return new Promise(resolve => {
                        mrshop.http.get('/user-center/user/checkValidCode',{
                            params:{
                                phone:args[0],
                                validcode:val
                            }
                        }).then(resp => {
                            resolve({
                                valid: resp.data.code == 200
                            })
                        }).catch(error => console.log(error));
                    });

                }
            })
```

3.5 完成注册



```
<div class="control-group">
    <label class="control-label"></label>
    <div class="controls btn-reg">
        <a class="sui-btn btn-block btn-xlarge btn-danger" href="javascript:void(0)"
          @click.stop="submit"
        >完成注册</a>
    </div>
</div>
```



```
    },
    submit() {
        this.$validator.validateAll().then(d => {
            if (d) {
                // 校验通过，提交表单
                mrshop.http.post("/user-center/user/register", this.user)
                    .then(resp => {
                        console.log(resp)
                        if (resp.data.code === 200) {
                            // 注册成功
                            alert("注册成功,即将跳转到登录页！");
                            setTimeout(() => window.location = '/login.html', 2000);
                        }
                    }).catch(() => alert("注册失败！"))
            }
        })
    }
```

```
                    this.$validator.validateAll().then(d => {
                        if (d) {
                            // 校验通过，提交表单
                            mrshop.http.post("/user-center/user/register",
this.user)
                                .then(resp => {
                                    console.log(resp)
                                    if (resp.data.code === 200) {
                                        // 注册成功
                                        alert("注册成功,即将跳转到登录页！");
                                        setTimeout(() => window.location =
'/login.html', 2000);
                                    }
                                }).catch(() => alert("注册失败！"))
                        }
                    })
```

4 用户登录功能实现

说明:想在这实现用户登录的话也可以,后面会讲到另一个登录相关的问题

5 其他模块

其他跟用户相关的还有好多模块,我就不一一给大家实现了,等大流程走完以后会给大家时间让大家自己做的