## 1.新增用户接口开发

### 1.1创建PasswordUtil工具类

**public class** PasswordUtil {  
  
 */\*\*  
 \* 定义password可以使用的字符  
 \*/* **public final static** String[] ***word*** = {  
 **"a"**, **"b"**, **"c"**, **"d"**, **"e"**, **"f"**, **"g"**,  
 **"h"**, **"j"**, **"k"**, **"m"**, **"n"**,  
 **"p"**, **"q"**, **"r"**, **"s"**, **"t"**,  
 **"u"**, **"v"**, **"w"**, **"x"**, **"y"**, **"z"**,  
 **"A"**, **"B"**, **"C"**, **"D"**, **"E"**, **"F"**, **"G"**,  
 **"H"**, **"J"**, **"K"**, **"M"**, **"N"**,  
 **"P"**, **"Q"**, **"R"**, **"S"**, **"T"**,  
 **"U"**, **"V"**, **"W"**, **"X"**, **"Y"**, **"Z"** };  
  
 */\*\*  
 \* 定义可以使用的数字  
 \*/* **public final static** String[] ***num*** = {  
 **"2"**, **"3"**, **"4"**, **"5"**, **"6"**, **"7"**, **"8"**, **"9"** };  
  
 */\*\*  
 \* 随机生成密码的方法  
 \*/* **public static** String randomPassword() {  
 StringBuffer stringBuffer = **new** StringBuffer();  
 *//使用单个 long 种子创建一个新的随机数生成器* Random random = **new** Random(System.*currentTimeMillis*());  
 *//每生成一个random时通过flag来标识选取数字还是字符* **boolean** flag = **false**;  
 *//* **int** length = random.nextInt(3) + 8;  
 **for** (**int** i = 0; i < length; i++) {  
 **if** (flag) {  
 stringBuffer.append(***num***[random.nextInt(***num***.**length**)]);  
 } **else** {  
 stringBuffer.append(***word***[random.nextInt(***word***.**length**)]);  
 }  
 flag = !flag;  
 }  
 **return** stringBuffer.toString();  
 }  
  
}

### 1.2创建SysUserService，新增用户的实现

*/\*\*  
 \* 新增用户的实现  
 \*  
 \** ***@param:*** *[param]  
 \** ***@return:*** *void  
 \** ***@author:*** *nino  
 \** ***@date:*** *2019/6/1 9:13  
 \*/***public void** save(UserParam param) {  
 BeanValidator.*check*(param);  
 **if**(checkTelephoneExist(param.getTelephone(), param.getId())) {  
 **throw new** ParamException(**"电话已被占用"**);  
 }  
 **if**(checkEmailExist(param.getMail(), param.getId())) {  
 **throw new** ParamException(**"邮箱已被占用"**);  
 }  
 String password = PasswordUtil.*randomPassword*();  
 *//* ***TODO:*** password = **"12345678"**;  
 String encryptedPassword = MD5Util.*encrypt*(password);  
 SysUser user = SysUser.*builder*()  
 .username(param.getUsername())  
 .telephone(param.getTelephone())  
 .mail(param.getMail())  
 .password(encryptedPassword)  
 .deptId(param.getDeptId())  
 .status(param.getStatus())  
 .remark(param.getRemark())  
 .build();  
 user.setOperator(RequestHolder.*getCurrentUser*().getUsername());  
 user.setOperateIp(IpUtil.*getRemoteIp*(RequestHolder.*getCurrentRequest*()));  
 user.setOperateTime(**new** Date());  
  
 *//* ***TODO: sendEmail* sysUserMapper**.insertSelective(user);  
 **sysLogService**.saveUserLog(**null**, user);  
}

*/\*\*  
 \* 检查邮箱是否已存在  
 \*  
 \** ***@param:*** *[mail, userId]  
 \** ***@return:*** *boolean  
 \** ***@author:*** *nino  
 \** ***@date:*** *2019/6/1 9:32  
 \*/***public boolean** checkEmailExist(String mail, Integer userId) {  
 **return sysUserMapper**.countByMail(mail, userId) > 0;  
}  
  
*/\*\*  
 \* 查看电话是否已存在  
 \*  
 \** ***@param:*** *[telephone, userId]  
 \** ***@return:*** *boolean  
 \** ***@author:*** *nino  
 \** ***@date:*** *2019/6/1 9:33  
 \*/***public boolean** checkTelephoneExist(String telephone, Integer userId) {  
 **return sysUserMapper**.countByTelephone(telephone, userId) > 0;  
}

SQL语句：

insertSelective对应的sql语句加入了NULL校验，即只会插入数据不为null的字段值。

<**insert id="insertSelective" parameterType="com.mmall.model.SysUser" keyProperty="id" useGeneratedKeys="true"**>  
 insert into sys\_user  
 <**trim prefix="(" suffix=")" suffixOverrides=","**>  
 <**if test="id != null"**>  
 id,  
 </**if**>  
 <**if test="username != null"**>  
 username,  
 </**if**>  
 <**if test="telephone != null"**>  
 telephone,  
 </**if**>  
 <**if test="mail != null"**>  
 mail,  
 </**if**>  
 <**if test="password != null"**>  
 password,  
 </**if**>  
 <**if test="deptId != null"**>  
 dept\_id,  
 </**if**>  
 <**if test="status != null"**>  
 status,  
 </**if**>  
 <**if test="remark != null"**>  
 remark,  
 </**if**>  
 <**if test="operator != null"**>  
 operator,  
 </**if**>  
 <**if test="operateTime != null"**>  
 operate\_time,  
 </**if**>  
 <**if test="operateIp != null"**>  
 operate\_ip,  
 </**if**>  
 </**trim**>  
 <**trim prefix="values (" suffix=")" suffixOverrides=","**>  
 <**if test="id != null"**>  
 #{id,jdbcType=INTEGER},  
 </**if**>  
 <**if test="username != null"**>  
 #{username,jdbcType=VARCHAR},  
 </**if**>  
 <**if test="telephone != null"**>  
 #{telephone,jdbcType=VARCHAR},  
 </**if**>  
 <**if test="mail != null"**>  
 #{mail,jdbcType=VARCHAR},  
 </**if**>  
 <**if test="password != null"**>  
 #{password,jdbcType=VARCHAR},  
 </**if**>  
 <**if test="deptId != null"**>  
 #{deptId,jdbcType=INTEGER},  
 </**if**>  
 <**if test="status != null"**>  
 #{status,jdbcType=INTEGER},  
 </**if**>  
 <**if test="remark != null"**>  
 #{remark,jdbcType=VARCHAR},  
 </**if**>  
 <**if test="operator != null"**>  
 #{operator,jdbcType=VARCHAR},  
 </**if**>  
 <**if test="operateTime != null"**>  
 #{operateTime,jdbcType=TIMESTAMP},  
 </**if**>  
 <**if test="operateIp != null"**>  
 #{operateIp,jdbcType=VARCHAR},  
 </**if**>  
 </**trim**>  
</**insert**>

*/\*\*  
 \* 新增用户请求  
 \*/*@RequestMapping(**"/save.json"**)  
@ResponseBody  
**public** JsonData saveUser(UserParam param) {  
 **sysUserService**.save(param);  
 **return** JsonData.*success*();  
}

### 1.3更新用户的实现

*/\*\*  
 \* 用户更新的实现，给后台用的  
 \*  
 \** ***@param:*** *[param]  
 \** ***@return:*** *void  
 \** ***@author:*** *nino  
 \** ***@date:*** *2019/6/1 10:38  
 \*/***public void** update(UserParam param) {  
 BeanValidator.*check*(param);  
 **if**(checkTelephoneExist(param.getTelephone(), param.getId())) {  
 **throw new** ParamException(**"电话已被占用"**);  
 }  
 **if**(checkEmailExist(param.getMail(), param.getId())) {  
 **throw new** ParamException(**"邮箱已被占用"**);  
 }  
 SysUser before = **sysUserMapper**.selectByPrimaryKey(param.getId());  
 Preconditions.*checkNotNull*(before, **"待更新的用户不存在"**);  
 SysUser after = SysUser.*builder*()  
 .id(param.getId())  
 .username(param.getUsername())  
 .telephone(param.getTelephone())  
 .mail(param.getMail())  
 .deptId(param.getDeptId())  
 .status(param.getStatus())  
 .remark(param.getRemark())  
 .build();  
 after.setOperator(RequestHolder.*getCurrentUser*().getUsername());  
 after.setOperateIp(IpUtil.*getRemoteIp*(RequestHolder.*getCurrentRequest*()));  
 after.setOperateTime(**new** Date());  
 **sysUserMapper**.updateByPrimaryKeySelective(after);  
 **sysLogService**.saveUserLog(before, after);  
}

SQL语句：

<**update id="updateByPrimaryKeySelective" parameterType="com.mmall.model.SysUser"**>  
 update sys\_user  
 <**set**>  
 <**if test="username != null"**>  
 username = #{username,jdbcType=VARCHAR},  
 </**if**>  
 <**if test="telephone != null"**>  
 telephone = #{telephone,jdbcType=VARCHAR},  
 </**if**>  
 <**if test="mail != null"**>  
 mail = #{mail,jdbcType=VARCHAR},  
 </**if**>  
 <**if test="password != null"**>  
 password = #{password,jdbcType=VARCHAR},  
 </**if**>  
 <**if test="deptId != null"**>  
 dept\_id = #{deptId,jdbcType=INTEGER},  
 </**if**>  
 <**if test="status != null"**>  
 status = #{status,jdbcType=INTEGER},  
 </**if**>  
 <**if test="remark != null"**>  
 remark = #{remark,jdbcType=VARCHAR},  
 </**if**>  
 <**if test="operator != null"**>  
 operator = #{operator,jdbcType=VARCHAR},  
 </**if**>  
 <**if test="operateTime != null"**>  
 operate\_time = #{operateTime,jdbcType=TIMESTAMP},  
 </**if**>  
 <**if test="operateIp != null"**>  
 operate\_ip = #{operateIp,jdbcType=VARCHAR},  
 </**if**>  
 </**set**>  
 where id = #{id,jdbcType=INTEGER}  
</**update**>

*/\*\*  
 \* 更新请求  
 \*/*@RequestMapping(**"/update.json"**)  
@ResponseBody  
**public** JsonData updateUser(UserParam param) {  
 **sysUserService**.update(param);  
 **return** JsonData.*success*();  
}