修改密码

初步分析：需要用户提交原始密码和新密码，再根据当前登录的用户进行信息的修改操作

1.修改密码-持久层

1.1规划需要执行的SQL语句

根据用户的uid修改用户password值

update t\_user set password=?,modified\_user=?, modified\_time=? WHERE uid=?

modified\_user=?, modified\_time=?是为了跟踪用户数据的变动,如果这条数据被错误修改了可以找到第一责任人

在执行修改密码之前，还应检查用户数据是否存在或者用户数据是否被标记为"已删除"(比如登录账号后的几分钟在和朋友聊天,没有看页面,管理员错误删除了你的账号或者错误设置is\_delete为1)、并检查原密码是否正确，这些检查都可以通过查询用户数据来辅助完成：

SELECT \* FROM t\_user WHERE uid=?

1.2设计接口和抽象方法

UserMapper接口,将以上的两个方法的抽象定义出来,将来映射到sql语句上

/\*\*

\* 根据用户的uid来修改用户密码

\* @param uid 用户的id

\* @param password 用户输入的新密码

\* @param modifiedUser 表示修改的执行者

\* @param modifiedTime 表示修改数据的时间

\* @return 返回值为受影响的行数

\*/

Integer updatePasswordByUid(Integer uid,

String password,

String modifiedUser,

Date modifiedTime);

/\*\*

\* 根据用户的id查询用户的数据

\* @param uid 用户的id

\* @return 如果找到则返回对象,反之返回null值

\*/

User findByUid(Integer uid);

1.3编写映射

配置到映射文件UserMapper.xml中

<update id="updatePasswordByUid">

update t\_user set

`password`=#{password},

modified\_user=#{modifiedUser},

modified\_time=#{modifiedTime},

where uid=#{uid}

</update>

<select id="findByUid" resultMap="UserEntityMap">

select \* from t\_user where uid=#{uid}

</select>

1.4单元测试

@Test

public void updatePasswordByUid(){

userMapper.updatePasswordByUid(

10,

"321",

"管理员",

new Date());

}

@Test

public void findByUid(){

System.out.println(userMapper.findByUid(10));

}

2.修改密码-业务层

2.1规划异常

用户的原密码错误,抛PasswordNotMatchException异常(前面已创建)

检测到is\_delete字段为1和uid找不到都是抛出用户没有找到的异常,UsernameNotFoundException(前面已创建)

update在更新的时候,有可能产生未知的异常,抛UpdateException异常

/\*\*用户在更新数据时产生的未知异常\*/

public class UpdateException extends ServiceException{

/\*\*重写ServiceException的所有构造方法\*/

}

2.2设计接口和抽象方法及实现

1.执行用户修改密码的核心方法:

/\*\*

\* changePassword方法需要什么参数:

\* 要先看底层持久层需要什么参数:uid,password,modifiedUser,modifiedTime

\* 1.修改人其实就是username,已经保存到session当中,通过控制层传递过来就行了

\* 2.在更新数据之前需要先根据uid查这个数据存不存在,uid也可以通过控制层传递

\* 3.新密码需要有

\* 4.修改时间不需要在参数列表,直接在方法内部new Date()就可以了

\* 5.旧密码

\* \*/

void changePassword(Integer uid,

String username,

String oldPassword,

String newPassword);

2.在实现类中实现当前的抽象方法

@Override

public void changePassword(Integer uid,

String username,

String oldPassword,

String newPassword) {

User result = userMapper.findByUid(uid);

/\*\*

\* 用户没找到:比如登录账号后的几分钟在和朋友聊天,没

\* 有看页面,管理员错误删除了你的账号或者错误设置is\_delete为1)

\*/

if (result ==null || result.getIsDelete() == 1) {

throw new UsernameNotFoundException("用户数据不存在");

}

//原始密码和数据库中密码进行比较

String oldMd5Password = getMD5Password(oldPassword,result.getSalt());

if (!result.getPassword().equals(oldMd5Password)) {

throw new PasswordNotMatchException("密码错误");

}

//将新的密码加密后设置到数据库中(只要曾经注册过就用以前的盐值)

String newMd5Password = getMD5Password(newPassword, result.getSalt());

Integer rows = userMapper.updatePasswordByUid(uid, newMd5Password, username, new Date());

if (rows != 1) {

throw new UpdateException("更新数据产生未知的异常");

}

}

2.3单元测试

@Test

public void changePassword() {

userService.changePassword(11,"管理员","123","321");

}

3.修改密码-控制层

3.1处理异常

UsernameNotFoundException异常和PasswordNotMatchException异常在前面的章节中已经处理过,现在只需要把UpdateException异常配置到统一的异常处理方法中

else if (e instanceof UpdateException) {

result.setState(5001);

result.setMessage("更新数据时产生未知的异常");

}

3.2设计请求

/users/change\_password

post

String oldPassword,String newPassword,HttpSession session(uid和username可以通过session获取到,在处理方法的内部获取就可以了)//如果参数名用的是非pojo类型,就需要和表单中的name属性值保持一致

JsonResult<void>

3.3处理请求

@RequestMapping("change\_password")

public JsonResult<Void> changePassword(String oldPassword,

String newPassword,

HttpSession session) {

Integer uid = getUidFromSession(session);

String username = getUsernameFromSession(session);

userService.changePassword(uid,username,oldPassword,newPassword);

return new JsonResult<>(OK);

}

启动服务,先登录账号然后在地址栏输入http://localhost:8080/users/change\_password?oldPassword=321&newPassword=123看看是否成功

4.修改密码-前端页面

在password.html中添加ajax请求的处理

<script>

$("#btn-change-password").click(function () {

$.ajax({

url: "/users/change\_password",

type: "POST",

data: $("#form-change-password").serialize(),

dataType: "JSON",

success: function (json) {

if (json.state == 200) {

alert("密码修改成功")

} else {

alert("密码修改失败")

}

},

error: function (xhr) {

//xhr.message可以获取未知异常的信息

alert("修改密码时产生未知的异常!"+xhr.message);

}

});

});

</script>

————————————————

版权声明：本文为CSDN博主「认真生活的灰太狼」的原创文章，遵循CC 4.0 BY-SA版权协议，转载请附上原文出处链接及本声明。

原文链接：https://blog.csdn.net/maxiangyu\_/article/details/124088948