单元测试说明书

一、测试内容

本系统的单元测试将对于 Service 模块的各个具体业务类进行逐一测试，包括 UserService，HouseService，NewsService，AuditService，SystemService，ReportService。每个业务模块均测试两个层次，分别为 Controller 层和 Service 层，其中 Controller 层的单元测试是为了测试后端方法对于前端传参的处理，Service 层的测试是为了测试具体的业务逻辑。

二、测试方法

1) 人工静态分析。后端程序员交叉查阅对方代码，检查有无逻辑错误。
2) 人工动态测试。由后端程序员设置输入参数和预计输出，然后核对工作交由测试程序去完成。

三、UserService 模块单元测试

3.1 Controller 层

3.1.1 测试描述

UserService 模块的 Controller 层单元测试主要是通过模拟 Http 请求，对 UserController 类的各个方法处理参数的能力做测试。

（1）测试对象

Controller 层的测试对象即为各个 Controller 类中的具体方法，详细描述如下表所示：

类名	UserController	
方法名	方法描述	参数说明
getUserList	获取用户列表	传入查询条件,包括用户名等
getUserDetail	获取用户详情	传入用户 id
manageRole	管理用户角色	传入用户角色 id
userEmpowerment	用户赋予角色	传入用户角色 id
getRoleList	获取角色列表	传入查询条件,包括角色名等
getRoleDetail	获取角色详情	传入角色 id
addRole	添加角色	传入角色信息,包括角色名等
getPermissionList	获取权限列表	传入查询条件,包括权限名等
addPermission	添加权限	传入权限信息
managePermission	管理角色权限	传入角色权限 id

（2）测试用例设计

Controller 层的职责为处理校验参数的合法性，合法参数可进行下一步操作，不合法参数便直接返回不合法信息，为了提高用户体验，对于输入的参数 Controller 层均能返回

HTTP200 的状态码，但非法信息会夹带自定义的非法状态码。所以测试用例的设计不能只顾于返回的 HTTP 状态码，还要具体校验返回的信息。

3.1.2 测试用例

对于每个测试对象都给出多个测试用例，以覆盖所有的参数输入情况，包括正常输入、边界值输入、空指输入等传参输入，并给出期望输出，和实例代码。

3.1.2.1 getUserList

（1）用例说明。其中参数均为 JSON 格式

序号	输入参数	期望结果	参数说明
US_UCON_1	{}	code:200	测试参数为 null 情况
US_UCON_2	{userId: 10000}	code:200	测试查询具体用户
US_UCON_3	{roleId:10000,orderText:”ur.CREATE_TIME asc”}	code:200	测试排序
US_UCON_4	{name: “张三”}	code:200	测试查询姓名
US_UCON_5	{pageSize:2}	code:200	测试分页

（2）测试代码实例

```
@ParameterizedTest
@ValueSource(strings = {"{\"userId\":10000}","{}","{\"roleId\":10000,\"orderText\":\"ur.CREATE_TIME
asc\"}","{\"name\":\"张三\"}","{\"pageSize\":2}"})
void getUserList(String arg) throws Exception {
    MvcResult mvcResult = null;
    try {
        mvcResult =    mockMvc.perform(MockMvcRequestBuilders.post("/user/getUserList")
                            .contentType(MediaType.APPLICATION_JSON)
                            .content(arg))
                .andExpect(MockMvcResultMatchers.status().isOk())
                .andDo(MockMvcResultHandlers.print())
                .andReturn();
        System.out.println(mvcResult);
    } catch (Exception e){
        e.printStackTrace();
    }
}
```

3.1.2.2 getUserDetail

（1）用例说明。其中参数均为 JSON 格式

序号	输入参数	期望结果	参数说明
US_UCON_6	{userId:10000}	code:200	测试正确传参
US_UCON_7	{}	code:500	测试传空
US_UCON_8	{roleId:10000}	code:500	测试不穿 userId

（2） 测试代码部分实例

```
@ParameterizedTest
@ValueSource(strings = {"{}","{\"roleId\":10000,\"orderText\":\"ur.CREATE_TIME asc\"}","{\"name\":\"张三\"}"})
void getUserDetail2(String arg){
    MvcResult mvcResult = null;
    try {
        mvcResult = mockMvc.perform(MockMvcRequestBuilders.post("/user/getUserDetail")
                                .contentType(MediaType.APPLICATION_JSON)
                                .content(arg))
                .andExpect(MockMvcResultMatchers.status().isOk())
                .andDo(MockMvcResultHandlers.print())
                .andReturn();
        Result<Map<String, Object>> result = JSONObject.parseObject(mvcResult.toString(), Result.class);
        int code = result.getCode();
        Assertions.assertEquals(500, code);
    } catch (Exception e){
        e.printStackTrace();
    }
}
```

3.1.2.3 manageRole

（1）用例说明。其中参数均为 JSON 格式

序号	输入参数	期望结果	参数说明
US_UCON_9	{userRoleId:10000, lockedMark:"NO"}	code:200	测试正确传参
US_UCON_10	{userRoleId:10000}	code:500	不传 lockedMark
US_UCON_11	{lockedMark:"NO"}	code:500	不穿 userRoleId
US_UCON_12	{userRoleId:10000, lockedMark:"DD"}	code:500	传入非法 lockedMark
US_UCON_13	{}	code:500	传空

（2） 测试代码部分实例

```
@ParameterizedTest
@ValueSource(strings =
{"{\"userRoleId\":10000}","{\"lockedMark\":\"NO\"}","{\"userId\":10000,\"lockedMark\":\"DD\"}"})
void manageRole2(String arg){
```

```
    MvcResult mvcResult = null;
    try {
        mvcResult = mockMvc.perform(MockMvcRequestBuilders.post("/user/manageRole")
                .contentType(MediaType.APPLICATION_JSON)
                .content(arg))
            .andExpect(MockMvcResultMatchers.status().isOk())
            .andDo(MockMvcResultHandlers.print())
            .andReturn();
        Result<Map<String, Object>> result = JSONObject.parseObject(mvcResult.toString(), Result.class);
        int code = result.getCode();
        Assertions.assertEquals(500, code);
    } catch (Exception e){
        e.printStackTrace();
    }
}
```

3.1.2.4 userEmpowerment

（1）用例说明。其中参数均为 JSON 格式

序号	输入参数	期望结果	参数说明
US_UCON_14	{roleId:10000,userId:10002}	code:200	测试正确传参
US_UCON_15	{roleId:10000}	code:500	不传 userId
US_UCON_16	{userId:10002}	code:500	不传 roleId

（2） 测试代码部分实例

```
@ParameterizedTest
@ValueSource(strings = {"{\"roleId\":\"10000","{\"userId\":10002}"})
void userEmpowerment2(String arg){
    MvcResult mvcResult = null;
    try {
        mvcResult = mockMvc.perform(MockMvcRequestBuilders.post("/user/userEmpowerment")
                .contentType(MediaType.APPLICATION_JSON)
                .content(arg))
            .andExpect(MockMvcResultMatchers.status().isOk())
            .andDo(MockMvcResultHandlers.print())
            .andReturn();
        Result<Map<String, Object>> result = JSONObject.parseObject(mvcResult.toString(), Result.class);
        int code = result.getCode();
        Assertions.assertEquals(500, code);
    } catch (Exception e){
        e.printStackTrace();
    }
}
```

3.1.2.5 getRoleList

（1）用例说明。其中参数均为 JSON 格式

序号	输入参数	期望结果	参数说明
US_UCON_17	{pageSize:5}	code:200	测试分页
US_UCON_18	{}	code:200	传空
US_UCON_19	{roleId:10004}	code:200	查询具体角色
US_UCON_20	{roleName:”管理”}	code:200	测试模糊查询

（2） 测试代码部分实例

```
@ParameterizedTest
@ValueSource(strings = {"{\"pageSize\":5}","{}","{\"roleId\":10004}","{\"pageNo\":1,\"roleName\":\"管理\"}"})
void getRoleList(String arg){
    MvcResult mvcResult = null;
    try {
        mvcResult = mockMvc.perform(MockMvcRequestBuilders.post("/user/getRoleList")
                .contentType(MediaType.APPLICATION_JSON)
                .content(arg))
            .andExpect(MockMvcResultMatchers.status().isOk())
            .andDo(MockMvcResultHandlers.print())
            .andReturn();
    } catch (Exception e){
        e.printStackTrace();
    }
}
```

3.1.2.6 getRoleDetail

（1）用例说明。其中参数均为 JSON 格式

序号	输入参数	期望结果	参数说明
US_UCON_21	{roleId:10000}	code:200	测试正确用例
US_UCON_22	{}	code:500	传空
US_UCON_23	{roleName:”管理员”}	code:500	roleId 传空

（2） 测试代码部分实例

```
@ParameterizedTest
@ValueSource(strings = {"{\"roleName\":\"管理员\"}"})
void getRoleDetail2(String arg){
    MvcResult mvcResult = null;
    try {
        mvcResult = mockMvc.perform(MockMvcRequestBuilders.post("/user/getRoleDetail")
                .contentType(MediaType.APPLICATION_JSON)
```

```
                .content(arg))
        .andExpect(MockMvcResultMatchers.status().isOk())
        .andDo(MockMvcResultHandlers.print())
        .andReturn();
    Result<Map<String, Object>> result = JSONObject.parseObject(mvcResult.toString(), Result.class);
    int code = result.getCode();
    Assertions.assertEquals(500, code);
  } catch (Exception e){
    e.printStackTrace();
  }
}
```

3.1.2.7 addRole

（1）用例说明。其中参数均为 JSON 格式

序号	输入参数	期望结果	参数说明
US_UCON_24	{roleName：“测试”，permissions:[10000,10001]}	code:200	测试正确用例
US_UCON_25	{}	code:500	传空
US_UCON_26	{roleName:”测试”}	code:200	权限传空
US_UCON_27	{permissions:[10000,10001]}	code:500	roleName 传空

（2） 测试代码部分实例

```
@ParameterizedTest
@ValueSource(strings = {"{\"roleId\":10000,\"permissions\":[10000]}"})
void addRole2(String arg){
  MvcResult mvcResult = null;
  try {
    mvcResult = mockMvc.perform(MockMvcRequestBuilders.post("/user/addRole")
                .contentType(MediaType.APPLICATION_JSON)
                .content(arg))
        .andExpect(MockMvcResultMatchers.status().isOk())
        .andDo(MockMvcResultHandlers.print())
        .andReturn();
    Result<Map<String, Object>> result = JSONObject.parseObject(mvcResult.toString(), Result.class);
    int code = result.getCode();
    Assertions.assertEquals(500, code);
  } catch (Exception e){
    e.printStackTrace();
  }
}
```

3.1.2.8 getPermissionList

（1）用例说明。其中参数均为 JSON 格式

序号	输入参数	期望结果	参数说明
US_UCON_28	{pageSize:10}	code:200	测试分页
US_UCON_29	{}	code:200	传空
US_UCON_30	{permissionId:10003}	code:200	查询具体权限
US_UCON_31	{url:”/user/test”}	code:200	查询 url

（2） 测试代码部分实例

```
@ParameterizedTest
@ValueSource(strings = {"{}","{\"pageSize\":10, \"pageNo\":1}","{\"permissionId\":10003}"})
void getPermissionList(String arg){
    MvcResult mvcResult = null;
    try {
        mvcResult = mockMvc.perform(MockMvcRequestBuilders.post("/user/getPermissionList")
                .contentType(MediaType.APPLICATION_JSON)
                .content(arg))
            .andExpect(MockMvcResultMatchers.status().isOk())
            .andDo(MockMvcResultHandlers.print())
            .andReturn();
    } catch (Exception e){
        e.printStackTrace();
    }
}
```

3.1.2.9 addPermission

（1）用例说明。其中参数均为 JSON 格式

序号	输入参数	期望结果	参数说明
US_UCON_32	{}	code:500	传空
US_UCON_33	{permissionName:”测试”}	code:500	不传 url
US_UCON_34	{url:”/user/test”}	code:500	不传权限名
US_UCON_35	{permissionName:” 测试”,url:”/user/test”}	code:200	正确用例

（2） 测试代码部分实例

```
@ParameterizedTest
@ValueSource(strings = {"{\"permissionName\":\"测试\", \"url\":\"\\user\\test\"}"})
void addPermission1(String arg){
    MvcResult mvcResult = null;
    try {
        mvcResult = mockMvc.perform(MockMvcRequestBuilders.post("/user/addPermission")
                .contentType(MediaType.APPLICATION_JSON)
                .content(arg))
            .andExpect(MockMvcResultMatchers.status().isOk())
            .andDo(MockMvcResultHandlers.print())
            .andReturn();
    } catch (Exception e){
        e.printStackTrace();
    }
}
```

3.1.2.10 managePermission

（1）用例说明。其中参数均为 JSON 格式

序号	输入参数	期望结果	参数说明
US_UCON_36	{}	code:500	传空
US_UCON_37	{rolePermissionId:10000}	code:500	不传 lockedMark
US_UCON_38	{lockedMark:"NO"}	code:500	不传角色权限 id
US_UCON_39	{rolePermissionId:10000,lockedMaek:"NO"}	code:200	正确用例

（2） 测试代码部分实例

```
@ParameterizedTest
@ValueSource(strings = {"{\"permissionId\":10000, \"lockedMark\":\"NO\"}"})
void managePermission1(String arg){
    MvcResult mvcResult = null;
    try {
        mvcResult = mockMvc.perform(MockMvcRequestBuilders.post("/user/manageRole")
                .contentType(MediaType.APPLICATION_JSON)
                .content(arg))
            .andExpect(MockMvcResultMatchers.status().isOk())
            .andDo(MockMvcResultHandlers.print())
            .andReturn();
    } catch (Exception e){
        e.printStackTrace();
```

```
    }
}
```

3.2 Service 层

3.2.1 测试描述

UserService 模块的 Service 层单元测试主要是通过传入测试人员设置好的参数，查看 Service 层各个函数的具体输出是否符合逻辑，比如返回的数量，返回的属性，是否会抛出异常等等。涉及到的 Service 类包括 UserService、RoleService、PermissionService。

（1）测试对象

Service 层的测试对象即为各个 Service 类中的具体方法，详细描述如下表所示：

类名	UserService	
方法名	方法描述	参数说明
selectUserList	获取用户列表	传入查询条件,包括用户名等
getUserDetail	获取用户详情	传入用户 id
manageRole	管理用户角色	传入用户角色 id
userEmpowerment	用户赋予角色	传入用户角色 id
getEmails	获取用户邮箱	传入用户 id
getModuleList	获取菜单栏列表	传入用户 id
类名	RoleService	
getRoleList	获取角色列表	传入查询条件,包括角色名等
getRoleDetail	获取角色详情	传入角色 id
addRole	添加角色	传入角色信息,包括角色名等
managePermission	管理角色权限	传入角色权限 id
类名	PermissionService	
getPermissionList	获取权限列表	传入查询条件,包括权限名等
addPermission	添加权限	传入权限信息

（2）测试用例设计

Service 层的职责在于处理具体的业务逻辑。其中 UserService 模块中的 Service 层主要是为了后台管理服务的，主要业务逻辑为数据库的增删改查，而对于算法没有什么要求，所以对于该 Service 层的单元测试主要是为了校验其返回值的主要属性，包括数量、主键是否正确，至于是否符合业务，需要到接口测试才能体现。

3.2.2 测试用例

对于每个测试对象都给出多个测试用例，以覆盖所有的参数输入情况，包括正常输入、边界值输入、空指输入等传参输入，并给出期望输出，和实例代码。

3.2.2.1 selectUserList

（1）用例说明。其中参数均为 UserForm 的封装属性。

序号	输入参数	期望结果	参数说明
US_USER_1	{}	非空	传空
US_USER_2	userId:10000	total:1,userId:10000	查询具体用户
US_USER_3	pageNo:1,pageSize:2	data.size:2	测试分页
US_UCON_4	roleId:10000	roleId:10000	测试具体查询条件

（2） 测试代码部分实例

```
@Test
void selectUserList(){
    UserForm userForm = new UserForm();
    Assertions.assertNotNull(userService.selectUserList(userForm).get("data"));
    userForm.setUserId(10000l);
    Assertions.assertEquals(1, (int)userService.selectUserList(userForm).get("total"));
    userForm = new UserForm();
    userForm.setPageNo(1);
    userForm.setPageSize(2);
    Assertions.assertEquals(2, ((List<UserVo>)userService.selectUserList(userForm).get("data")).size());
}
```

3.2.2.2 getUserDetail

（1）用例说明。其中参数均为 UserForm 的封装属性。

序号	输入参数	期望结果	参数说明
US_USER_5	userId:10000	userId:10000	测试正确用例
US_USER_6		flag:false	传空

（2） 测试代码部分实例

```
@Test
void getUserDetail(){
    UserForm userForm = new UserForm();
    userForm.setUserId(10000l);
    Map<String, Object> result = userService.getUserDetail(userForm);
    UserVo userVo = (UserVo) result.get("data");
    Assertions.assertEquals(10000l, userVo.getUserId());
}
```

3.2.2.3 manageRole

（1）用例说明。其中参数均为 UserForm 的封装属性。

序号	输入参数	期望结果	参数说明

US_USER_7	userRoleId:1,lockedMark:”NO”	flag:true	正确用例
US_USER_8		flag:flase	传空
US_USER_9	userRoleId:1	flag:false	lockedMark 传空
US_USER_10	lockedMark:”NO”	flag:false	userRoleId 传空

（2） 测试代码部分实例

```
@Test
void manageRole(){
    UserForm userForm = new UserForm();
    userForm.setUserRoleId(2l);
    userForm.setLockedMark("NO");
    Boolean flag = (Boolean) userService.manageRole(userForm).get("flag");
    Assertions.assertEquals(true, flag);
    userForm = new UserForm();
    userForm.setUserId(2l);
    flag = (Boolean) userService.manageRole(userForm).get("flag");
    Assertions.assertEquals(false, flag);
    userForm = new UserForm();
    userForm.setLockedMark("NO");
    flag = (Boolean) userService.manageRole(userForm).get("flag");
    Assertions.assertEquals(false, flag);
}
```

3.2.2.4 userEmpowerment

（1）用例说明。其中参数均为 UserForm 的封装属性。

序号	输入参数	期望结果	参数说明
US_USER_11	userId:10002,roleId:10001	flag:true	正确用例
US_USER_12	userId:10000,roleId:10000	flag:flase	重复赋予角色
US_USER_13		flag:false	传空
US_USER_14	userId:10002	flag:false	roleId 传空
US_USER_15	roleId:10003	flag:false	userId 传空

（2） 测试代码部分实例

```
@Test
void userEmpowerment(){
    UserForm userForm = new UserForm();
    userForm.setUserId(10002l);
    userForm.setRoleId(10000);
    Boolean flag = (Boolean) userService.userEmpowerment(userForm).get("flag");
    Assertions.assertEquals(true, flag);
    userForm = new UserForm();
```

```
    userForm.setUserId(10002l);
    flag = (Boolean) userService.userEmpowerment(userForm).get("flag");
    Assertions.assertEquals(false, flag);
    userForm = new UserForm();
    userForm.setRoleId(10000);
    flag = (Boolean) userService.userEmpowerment(userForm).get("flag");
    Assertions.assertEquals(false, flag);
    userForm = new UserForm();
    userForm.setUserId(10002l);
    userForm.setRoleId(10001);
    flag = (Boolean) userService.userEmpowerment(userForm).get("flag");
    Assertions.assertEquals(false, flag);
}
```

3.2.2.5 getEmails

（1）用例说明。

序号	输入参数	期望结果	参数说明
US_USER_16	userIds:[10000,10001]	非空	正确用例
US_USER_17		null	传空

（2） 测试代码部分实例

```
@Test
void getEmails(){
    List<Long> userIds = new ArrayList<>();
    userIds.add(10000l);
    userIds.add(10001l);
    List<String> emails = (List<String>) userService.getEmail(userIds).get("data");
    Assertions.assertNotNull(emails);
    userIds = new ArrayList<>();
    emails = (List<String>) userService.getEmail(userIds).get("data");
    Assertions.assertEquals(null, emails);
}
```

3.2.2.6 getModuleList

（1）用例说明。其中参数均为 UserForm 的封装属性。

序号	输入参数	期望结果	参数说明
US_USER_18	userId:10000	非空	正确用例
US_USER_19		null	传空

（2） 测试代码部分实例

```
@Test
void getModuleList(){
    Long userId = 10000l;
    Assertions.assertNotNull(userService.getModuleList(userId));
}
```

3.2.2.7 getRoleList

（1）用例说明。其中参数均为 RoleForm 的封装属性。

序号	输入参数	期望结果	参数说明
US_USER_20		非空	传空
US_USER_21	roleId:10000	total:1,roleId:10000	查询某个角色
US_USER_22	pageSize:2,pageNo:1	data.size:2	测试分页

（2） 测试代码部分实例

```
@Test
void getRoleList(){
    RoleForm roleForm = new RoleForm();
    List<RoleVo> list = (List<RoleVo>) roleService.getRoleList(roleForm).get("data");
    Assertions.assertNotNull(list);
    roleForm.setRoleId(10000);
    Assertions.assertEquals(1, (int)roleService.getRoleList(roleForm).get("total"));
    roleForm = new RoleForm();
    roleForm.setPageSize(2);
    Assertions.assertEquals(2, ((List<RoleVo>) roleService.getRoleList(roleForm)).size());
}
```

3.2.2.8 getRoleDetail

（1）用例说明。其中参数均为 RoleForm 的封装属性。

序号	输入参数	期望结果	参数说明
US_USER_23		flag:false	传空
US_USER_24	roleId:10000	roleId:10000	正确用例

（2） 测试代码部分实例

```
@Test
void getRoleDetail(){
    RoleForm roleForm = new RoleForm();
    roleForm.setRoleId(10001);
    Map<String, Object> result = roleService.getRoleDetail(roleForm);
    RoleVo roleVo = (RoleVo) result.get("data");
```

```
    Assertions.assertEquals(10001, roleVo.getRoleId());
}
```

3.2.2.9 managePermission

（1）用例说明。其中参数均为 RoleForm 的封装属性。

序号	输入参数	期望结果	参数说明
US_USER_25		flag:false	传空
US_USER_26	rolePermissionId:10000	flag:false	lockedMark 传空
US_USER_27	lockedMark:”NO”	flag:false	rolePermissionId 传空
US_USER_28	rolePermissionId:10000,lockedMark:”NO”	flag:true	正确用例

（2） 测试代码部分实例

```
@Test
void managePermission(){
    RoleForm roleForm = new RoleForm();
    roleForm.setRolePermissionId(10003);
    roleForm.setLockedMark("NO");
    Boolean flag = (Boolean) roleService.managePermission(roleForm).get("flag");
    Assertions.assertEquals(true, flag);
    roleForm = new RoleForm();
    roleForm.setLockedMark("NO");
    flag = (Boolean) roleService.managePermission(roleForm).get("flag");
    Assertions.assertEquals(false, flag);
    roleForm = new RoleForm();
    roleForm.setRolePermissionId(10003);
    flag = (Boolean) roleService.managePermission(roleForm).get("flag");
    Assertions.assertEquals(false, flag);
}
```

3.2.2.10 addRole

（1）用例说明。其中参数均为 RoleForm 的封装属性。

序号	输入参数	期望结果	参数说明
US_USER_29		flag:false	传空
US_USER_30	roleName:”测试”	flag:true,roleId 非空	permissions 传空
US_USER_31	roleName:”测试”, permissions:[10000,10001]	flag:true,roleId 非空	正确用例

（2） 测试代码部分实例

```
@Test
void addRole(){
    RoleForm roleForm = new RoleForm();
    Boolean flag = (Boolean) roleService.addRole(roleForm).get("flag");
    Assertions.assertEquals(false, flag);
    roleForm.setRoleName("测试");
    Map<String, Object> result = roleService.addRole(roleForm);
    flag = (Boolean) result.get("flag");
    Assertions.assertEquals(true, flag);
    Assertions.assertNotNull(result.get("roleId"));
    roleForm = new RoleForm();
    roleForm.setRoleName("测试");
    List<Integer> list = new ArrayList<>();
    list.add(10000);
    list.add(10001);
    roleForm.setPermissions(list);
    flag = (Boolean) roleService.addRole(roleForm).get("flag");
    Assertions.assertEquals(true, flag);
}
```

3.2.2.11 getPermissionList

（1）用例说明。其中参数均为 PermissionForm 的封装属性。

序号	输入参数	期望结果	参数说明
US_USER_32		非空	传空
US_USER_33	pageSize:3,pageNo:2	data.size:3,total != 3	测试分页
US_USER_34	permissionId:10000	total:1,permissionId:10000	查询具体权限

（2） 测试代码部分实例

```
@Test
void getPermissionList(){
    PermissionForm permissionForm = new PermissionForm();
    Map<String, Object> result = permissionService.getPermissionList(permissionForm);
    List<PermissionVo> list = (List<PermissionVo>) result.get("data");
    Assertions.assertNotNull(list);
    permissionForm.setPageSize(3);
    result = permissionService.getPermissionList(permissionForm);
    list = (List<PermissionVo>) result.get("data");
    Assertions.assertEquals(3, list.size());
    permissionForm.setPermissionId(10000);
    Assertions.assertEquals(1, (int) permissionService.getPermissionList(permissionForm).get("total"));
}
```

3.2.2.12 addPermission

（1）用例说明。其中参数均为 PermissionForm 的封装属性。

序号	输入参数	期望结果	参数说明
US_USER_35		flag:false	传空
US_USER_36	permissonName:”测试”	flag:false	url 为空
US_USER_37	url:”/user/test”	flag:false	权限名为空
US_USER_38	permissionName:” 测试”,url:”/user/test”	flag:true,permissionId 非空	正确用例

（2） 测试代码部分实例

```
@Test
void addPermission(){
    PermissionForm permissionForm = new PermissionForm();
    permissionForm.setPermissionName("测试");
    permissionForm.setUrl("/user/test");
    Boolean flag = (Boolean) permissionService.addPermission(permissionForm).get("flag");
    Assertions.assertEquals(true, flag);
    permissionForm = new PermissionForm();
    permissionForm.setUrl("/user/test");
    flag = (Boolean) permissionService.addPermission(permissionForm).get("flag");
    Assertions.assertEquals(false, flag);
    permissionForm = new PermissionForm();
    permissionForm.setPermissionName("测试");
    flag = (Boolean) permissionService.addPermission(permissionForm).get("flag");
    Assertions.assertEquals(false, flag);
}
```

四、ReportService 模块单元测试

4.1 Controller 层

4.1.1 测试描述

ReportService 模块的 Controller 层单元测试主要是通过模拟 Http 请求，对 UserController 类的各个方法处理参数的能力做测试。

（1）测试对象

Controller 层的测试对象即为各个 Controller 类中的具体方法，详细描述如下表所示：

类名	ReportController

方法名	方法描述	参数说明
addReport	添加举报记录	传入举报基本信息
getReportList	获取举报记录	传入查询信息比如类别 id 等
getReportDetail	获取举报详情	传入举报 id
doAudit	举报审核	传入举报和审核信息

（2）测试用例设计

Controller 层的职责为处理校验参数的合法性，合法参数可进行下一步操作，不合法参数便直接返回不合法信息，为了提高用户体验，对于输入的参数 Controller 层均能返回 HTTP200 的状态码，但非法信息会夹带自定义的非法状态码。所以测试用例的设计不能只顾于返回的 HTTP 状态码，还要具体校验返回的信息。

4.1.2 测试用例

对于每个测试对象都给出多个测试用例，以覆盖所有的参数输入情况，包括正常输入、边界值输入、空指输入等传参输入，并给出期望输出，和实例代码。

4.1.2.1 selectReportList

（1）用例说明。其中参数均为 ReportForm 的封装属性。

序号	输入参数	期望结果	参数说明
RS_RCON_1	{}	code:200	传空
RS_RCON_2	{reportId:10000}	code:200	查询具体举报
RS_RCON_3	{pageNo:1,pageSize:2}	code:200	测试分页
RS_RCON_4	{orderText:”CREATE_TIME desc”}	code:200	测试排序

（2） 测试代码部分实例

```
@ParameterizedTest
@ValueSource(strings = {"{\"objtypeId\":10000}"})
void testGetReportList(String arg) throws Exception {
    // Setup
    when(mockReportService.getReportList(new ReportForm())).thenReturn(new HashMap<>());

    // Run the test
    final MockHttpServletResponse response = mockMvc.perform(get("/report/getReportList")
                    .content(arg).contentType(MediaType.APPLICATION_JSON)
                    .accept(MediaType.APPLICATION_JSON))
            .andReturn().getResponse();

    // Verify the results
    assertThat(response.getStatus()).isEqualTo(HttpStatus.OK.value());
}
```

4.1.2.2 addReport

（1）用例说明。其中参数均为 ReportForm 的封装属性。

序号	输入参数	期望结果	参数说明
RS_RCON_5	{}	code:500	传空
RS_RCON_6	{defense:10000, reason:”xx”}	code:500	举报类型为空
RS_RCON_7	{defense:10000, objtypeId:10000}	code:500	举报理由为空
RS_RCON_8	{reason:”xx”, objtypeId:10000}	code:500	被举报对象为空
RS_RCON_9	{defense:10000,reason:”xx”,objtypeId:10000}	code:200	正确用例

（2） 测试代码部分实例

```
@Test
void testAddReport() throws Exception {
    // Setup
    when(mockReportService.addReport(new ReportForm())).thenReturn(new HashMap<>());

    // Run the test
    final MockHttpServletResponse response = mockMvc.perform(post("/report/addReport")
                        .content("{}").contentType(MediaType.APPLICATION_JSON)
                        .accept(MediaType.APPLICATION_JSON))
            .andReturn().getResponse();

    // Verify the results
    assertThat(response.getStatus()).isEqualTo(HttpStatus.OK.value());
    assertThat(response.getContentAsString()).isEqualTo("expectedResponse");
}
```

4.1.2.3 getReportDetail

（1）用例说明。其中参数均为 ReportForm 的封装属性。

序号	输入参数	期望结果	参数说明
RS_RCON_10	{}	code:500	传空
RS_RCON_11	{reportId:10000}	code:200	正确用例

（2） 测试代码部分实例

```
@ParameterizedTest
@ValueSource(strings = {"{\"objtypeId\":10000}"})
void testGetReportDetail(String arg) throws Exception {
    // Setup
    when(mockReportService.getReportDetail(new ReportForm())).thenReturn(new HashMap<>());
```

```
    // Run the test
    final MockHttpServletResponse response = mockMvc.perform(get("/report/getReportDetail")
                        .content(arg).contentType(MediaType.APPLICATION_JSON)
                        .accept(MediaType.APPLICATION_JSON))
                .andReturn().getResponse();

    // Verify the results
    assertThat(response.getStatus()).isEqualTo(HttpStatus.OK.value());
    assertThat(response.getContentAsString()).isEqualTo("expectedResponse");
}
```

4.1.2.4 doAudit

（1）用例说明。其中参数均为 ReportForm 的封装属性。

序号	输入参数	期望结果	参数说明
RS_RCON_12	{objtypeId:10000, defense:10000}	code:200	正确用例
RS_RCON_13	{}	code:500	传空
RS_RCON_14	{objtypeId:10000}	code:500	defense 传空
RS_RCON_15	{defense:10000}	code:500	objtypeId 传空

（2） 测试代码部分实例

```
@Test
void testDoAudit() throws Exception {
    // Setup
    when(mockReportService.doAudit(new ReportForm())).thenReturn(false);
    // Run the test
    final MockHttpServletResponse response = mockMvc.perform(post("/report/doAudit")
                        .content("content").contentType(MediaType.APPLICATION_JSON)
                        .accept(MediaType.APPLICATION_JSON))
                .andReturn().getResponse();
    // Verify the results
    assertThat(response.getStatus()).isEqualTo(HttpStatus.OK.value());
    assertThat(response.getContentAsString()).isEqualTo("expectedResponse");
}
```

4.2 Service 层

4.2.1 测试描述

ReportService 模块的 Service 层单元测试主要是通过传入测试人员设置好的参数，查看 Service 层各个函数的具体输出是否符合逻辑，比如返回的数量，返回的属性，是否会抛出异常等等。涉及到的 Service 类包括 ReportService。

（1）测试对象

Service 层的测试对象即为各个 Service 类中的具体方法，详细描述如下表所示：

类名	ReportService	
方法名	方法描述	参数说明
addReport	添加举报记录	传入举报基本信息
getReportList	获取举报记录	传入查询信息比如类别 id 等
getReportDetail	获取举报详情	传入举报 id
doAudit	举报审核	传入举报和审核信息

（2）测试用例设计

Service 层的职责在于处理具体的业务逻辑。其中 ReportService 模块中的 Service 层主要是为了后台管理服务的，主要业务逻辑为数据库的增删改查，而对于算法没有什么要求，所以对于该 Service 层的单元测试主要是为了校验其返回值的主要属性，包括数量、主键是否正确，至于是否符合业务，需要到接口测试才能体现。

4.2.2 测试用例

对于每个测试对象都给出多个测试用例，以覆盖所有的参数输入情况，包括正常输入、边界值输入、空指输入等传参输入，并给出期望输出，和实例代码。

4.2.2.1 getReportList

（1）用例说明。其中参数均为 ReportForm 的封装属性。

序号	输入参数	期望结果	参数说明
RS_RSER_1		非空	传空
RS_RSER_2	pageSize:3,pageNo:1	data.size:3	测试分页
RS_RSER_3	objtypeId:10000,defense:10000,prosecution:10000	total:1	查询具体举报
RS_RSER_4	isHandle:”YES”	非空	查询已处理的举报

（2） 测试代码部分实例

```
@Test
void testGetReportList() {
    // Setup
    final ReportForm reportForm = new ReportForm();
    reportForm.setReportId(0);
    reportForm.setCreateTime(Date.valueOf(LocalDate.of(2020, 1, 1)));
    reportForm.setObjtypeId(0);
    reportForm.setDefense(0L);
    reportForm.setReason("reason");
    reportForm.setProsecution(0L);
    reportForm.setHandleMark("handleMark");
    reportForm.setDeleteMark("deleteMark");
```

```
    reportForm.setPageSize(0);
    reportForm.setPageNo(0);

    // Run the test
    final Map<String, Object> result = reportServiceImplUnderTest.getReportList(reportForm);

    // Verify the results
}
```

4.2.2.2 getReportDetail

（1）用例说明。其中参数均为 ReportForm 的封装属性。

序号	输入参数	期望结果	参数说明
RS_RSER_5		null	传空
RS_RSER_6	pageSize:3,pageNo:1	data.size:3	测试分页
RS_RSER_7	objtypeId:10000,defense:10000	非空	查询详情
RS_RSER_8	isHandle:”YES”	null	查询已处理的举报

（2） 测试代码部分实例

```
@Override
public Map<String, Object> getReportDetail(ReportForm reportForm) {
    Map<String, Object> params = new HashMap<>();
    params.put("start", (reportForm.getPageNo() - 1) * reportForm.getPageSize());
    params.put("size", reportForm.getPageSize());
    params.put("objtypeId", reportForm.getObjtypeId());
    params.put("objId", reportForm.getDefense());

    Map<String, Object> result = new HashMap<>();

    List<ReportVo> list = new ArrayList<>();

    Integer total = baseMapper.countReportDetail(params);
    if(total > 0){
        list = baseMapper.getReportList(params);
    }

    result.put("total", total);
    result.put("data", list);
    result.put("pageNo", reportForm.getPageNo());
    result.put("pageSize", reportForm.getPageSize());

    return result;
}
```

4.2.2.3 addReport

（1）用例说明。其中参数均为 ReportForm 的封装属性。

序号	输入参数	期望结果	参数说明
RS_RSER_9		flag:false	传空
RS_RSER_10	defense:10000,reason:”xx”,objtypeId:10000,prosecution:10000	flag:true,reportId:非空	正确用例
RS_RSER_11	objtypeId:10000,defense:10000	flag:false	理由和举报人为空

（2） 测试代码部分实例

```
@Test
void testAddReport() {
    // Setup
    final ReportForm reportForm = new ReportForm();
    reportForm.setReportId(0);
    reportForm.setCreateTime(Date.valueOf(LocalDate.of(2020, 1, 1)));
    reportForm.setObjtypeId(0);
    reportForm.setDefense(0L);
    reportForm.setReason("reason");
    reportForm.setProsecution(0L);
    reportForm.setHandleMark("handleMark");
    reportForm.setDeleteMark("deleteMark");
    reportForm.setPageSize(0);
    reportForm.setPageNo(0);

    // Run the test
    final Map<String, Object> result = reportServiceImplUnderTest.addReport(reportForm);

    // Verify the results
}
```

4.2.2.3 doAudit

（1）用例说明。其中参数均为 ReportForm 的封装属性。

序号	输入参数	期望结果	参数说明
RS_RSER_12		flag:false	传空
RS_RSER_13	defense:10000,objtypeId:10000	flag:true,update:非空	正确用例

（2） 测试代码部分实例

```
@Test
void testDoAudit() {
```

```
    // Setup
    final ReportForm reportForm = new ReportForm();
    reportForm.setReportId(0);
    reportForm.setCreateTime(Date.valueOf(LocalDate.of(2020, 1, 1)));
    reportForm.setObjtypeId(0);
    reportForm.setDefense(0L);
    reportForm.setReason("reason");
    reportForm.setProsecution(0L);
    reportForm.setHandleMark("handleMark");
    reportForm.setDeleteMark("deleteMark");
    reportForm.setPageSize(0);
    reportForm.setPageNo(0);

    when(mockReportDao.updateIsHandle(Arrays.asList(new Report()))).thenReturn(0);

    // Run the test
    final Boolean result = reportServiceImplUnderTest.doAudit(reportForm);

    // Verify the results
    assertThat(result).isTrue();
}
```

五、AuditService 模块单元测试

5.1 Controller

5.1.1 测试描述

AuditService 模块的 Controller 层单元测试主要是通过模拟 Http 请求，对 AuditController 类的各个方法处理参数的能力做测试。

（1）测试对象

Controller 层的测试对象即为各个 Controller 类中的具体方法，详细描述如下表所示：

类名	AuditController	
方法名	方法描述	参数说明
getAuditList	获取审核列表	传入查询信息包括审核 id 等
addAudit	添加审核记录	传入查询信息比如类别 id 等

（2）测试用例设计

Controller 层的职责为处理校验参数的合法性，合法参数可进行下一步操作，不合法参数便直接返回不合法信息，为了提高用户体验，对于输入的参数 Controller 层均能返回 HTTP200 的状态码，但非法信息会夹带自定义的非法状态码。所以测试用例的设计不能只顾于返回的 HTTP 状态码，还要具体校验返回的信息。

5.1.2 测试用例

对于每个测试对象都给出多个测试用例，以覆盖所有的参数输入情况，包括正常输入、边界值输入、空指输入等传参输入，并给出期望输出，和实例代码。

5.1.2.1 getAuditList

（1）用例说明。其中参数均为 AuditForm 的封装属性。

序号	输入参数	期望结果	参数说明
AS_ACON_1	{}	code:200	传空
AS_ACON_2	{auditId:10000}	code:200	查询具体举报
AS_ACON_3	{pageNo:1,pageSize:2}	code:200	测试分页
AS_ACON_4	{orderText:"CREATE_TIME desc"}	code:200	测试排序

（2） 测试代码部分实例

```
@Test
void testGetAuditList() throws Exception {
    // Setup
    // Configure AuditService.getAuditList(...).
    final AuditVo auditVo = new AuditVo();
    auditVo.setAuditId(0);
    auditVo.setCreateTime(Date.valueOf(LocalDate.of(2020, 1, 1)));
    auditVo.setObjtypeId(0);
    auditVo.setObjId(0L);
    auditVo.setOperator(0L);
    auditVo.setOper(0);
    auditVo.setOperation("operation");
    auditVo.setMessage("message");
    auditVo.setDeleteMark("deleteMark");
    final List<AuditVo> auditVos = Arrays.asList(auditVo);
    when(mockAuditService.getAuditList(new AuditForm())).thenReturn(auditVos);

    // Run the test
    final MockHttpServletResponse response = mockMvc.perform(get("/audit/getAuditList")
                    .content("content").contentType(MediaType.APPLICATION_JSON)
                    .accept(MediaType.APPLICATION_JSON))
            .andReturn().getResponse();

    // Verify the results
    assertThat(response.getStatus()).isEqualTo(HttpStatus.OK.value());
    assertThat(response.getContentAsString()).isEqualTo("expectedResponse");
}
```

5.1.2.2 addAudit

（1）用例说明。其中参数均为 AuditForm 的封装属性。

序号	输入参数	期望结果	参数说明
AS_ACON_5	{}	code:500	传空
AS_ACON_6	{objtypeId:10000,objId:10000,oper:10000}	code:200	正确用例
AS_ACON_7	{objtypeId:10000,objId:10000}	code:500	操作为空

（2） 测试代码部分实例

```
@Test
void testAddAudit() throws Exception {
    // Setup
    when(mockAuditService.addAudit(new AuditForm())).thenReturn(new HashMap<>());

    // Run the test
    final MockHttpServletResponse response = mockMvc.perform(post("/audit/addAudit")
                        .content("content").contentType(MediaType.APPLICATION_JSON)
                        .accept(MediaType.APPLICATION_JSON))
            .andReturn().getResponse();

    // Verify the results
    assertThat(response.getStatus()).isEqualTo(HttpStatus.OK.value());
    assertThat(response.getContentAsString()).isEqualTo("expectedResponse");
}
```

5.2 Service 层

5.2.1 测试描述

AuditService 模块的 Service 层单元测试主要是通过传入测试人员设置好的参数，查看 Service 层各个函数的具体输出是否符合逻辑，比如返回的数量，返回的属性，是否会抛出异常等等。涉及到的 Service 类包括 AuditService。

（1）测试对象

Service 层的测试对象即为各个 Service 类中的具体方法，详细描述如下表所示：

类名	AuditService	
方法名	方法描述	参数说明
addAudit	添加审核记录	传入审核信息
getAuditList	获取审核记录	传入查询信息比如类别 id 等

（2）测试用例设计

Service 层的职责在于处理具体的业务逻辑。其中 AuditService 模块中的 Service 层主要是为了后台管理服务的，主要业务逻辑为数据库的增删改查，而对于算法没有什么要求，所以对于该 Service 层的单元测试主要是为了校验其返回值的主要属性，包括数量、主键是否正确，至于是否符合业务，需要到接口测试才能体现。

5.2.2 测试用例

对于每个测试对象都给出多个测试用例，以覆盖所有的参数输入情况，包括正常输入、边界值输入、空指输入等传参输入，并给出期望输出，和实例代码。

5.2.2.1 addAudit

（1）用例说明。其中参数均为 AuditForm 的封装属性。

序号	输入参数	期望结果	参数说明
AS_ASER_1		flag:false	传空
AS_ASER_2	objtypeId:10000,objId:10000,oper:10000	flag:true,auditId:非空	正确用例
AS_ASER_3	objtypeId:10000,objId:10000	flag:false	操作为空

（2） 测试代码部分实例

```
@Test
void testAddAudit() {
    // Setup
    final AuditForm auditForm = new AuditForm();
    auditForm.setAuditId(0);
    auditForm.setCreateTime(Date.valueOf(LocalDate.of(2020, 1, 1)));
    auditForm.setObjtypeId(0);
    auditForm.setObjId(0L);
    auditForm.setStatus(0);
    auditForm.setOperator(0L);
    auditForm.setOper(0);
    auditForm.setMessage("message");
    auditForm.setDeleteMark("deleteMark");

    // Run the test
    final Map<String, Object> result = auditServiceImplUnderTest.addAudit(auditForm);

    // Verify the results
}
```

5.2.2.2 getAuditList

（1）用例说明。其中参数均为 AuditForm 的封装属性。

序号	输入参数	期望结果	参数说明
AS_ASER_4		非空	传空
AS_ASER_5	pageSize:3,pageNo:1	data.size:3	测试分页
AS_ASER_6	objtypeId:10000,objId:10000	非空	测试查询条件
AS_ASER_7	orderText:CREATE_TIME desc	非空	测试排序
AS_ASER_8	auditId:10000	total:1	插叙具体审核

（2） 测试代码部分实例

```
@Test
void testGetAuditList() {
    // Setup
    final AuditForm auditForm = new AuditForm();
    auditForm.setAuditId(0);
    auditForm.setCreateTime(Date.valueOf(LocalDate.of(2020, 1, 1)));
    auditForm.setObjtypeId(0);
    auditForm.setObjId(0L);
    auditForm.setStatus(0);
    auditForm.setOperator(0L);
    auditForm.setOper(0);
    auditForm.setMessage("message");
    auditForm.setDeleteMark("deleteMark");

    final AuditVo auditVo = new AuditVo();
    auditVo.setAuditId(0);
    auditVo.setCreateTime(Date.valueOf(LocalDate.of(2020, 1, 1)));
    auditVo.setObjtypeId(0);
    auditVo.setObjId(0L);
    auditVo.setOperator(0L);
    auditVo.setOper(0);
    auditVo.setOperation("operation");
    auditVo.setMessage("message");
    auditVo.setDeleteMark("deleteMark");
    final List<AuditVo> expectedResult = Arrays.asList(auditVo);

    // Run the test
    final List<AuditVo> result = auditServiceImplUnderTest.getAuditList(auditForm);

    // Verify the results
    assertThat(result).isEqualTo(expectedResult);
}
```

六、SystemService 模块单元测试

6.1 Controller

6.1.1 测试描述

SystemService 模块的 Controller 层单元测试主要是通过模拟 Http 请求，对 SyslogController、MessageController 类的各个方法处理参数的能力做测试。

（1）测试对象

Controller 层的测试对象即为各个 Controller 类中的具体方法，详细描述如下表所示：

类名	MessageController	
方法名	方法描述	参数说明
launchMessage	发送信息	传入信息基本内容
类名	SyslogController	
方法名	方法描述	参数说明
addLog	添加日志	传入日志内容

（2）测试用例设计

Controller 层的职责为处理校验参数的合法性，合法参数可进行下一步操作，不合法参数便直接返回不合法信息，为了提高用户体验，对于输入的参数 Controller 层均能返回 HTTP200 的状态码，但非法信息会夹带自定义的非法状态码。所以测试用例的设计不能只顾于返回的 HTTP 状态码，还要具体校验返回的信息。

6.1.2 测试用例

对于每个测试对象都给出多个测试用例，以覆盖所有的参数输入情况，包括正常输入、边界值输入、空指输入等传参输入，并给出期望输出，和实例代码。

6.1.2.1 launchMessage

（1）用例说明。其中参数均为 MessageForm 的封装属性。

序号	输入参数	期望结果	参数说明
SS_SCON_1		code:500	传空
SS_SCON_2	{scope:10000, content:”xxxx”}	code:500	接收者为空
SS_SCON_3	{content:”xxx”,specificUsers:[10000, 10001]}	code:500	范围为空
SS_SCON_4	{scope:10000,specificUsers:[10000,10001]}	code:500	内容为空
SS_SCON_5	{scope:10000,specificUsers:[10000,10001],content:”xxx”}	code:200	正确用例

（2） 测试代码部分实例

```
@Test
void testLaunchMessage() throws Exception {
```

```
    // Setup
    when(mockMessageService.launchMessage(new MessageForm())).thenReturn(new HashMap<>());
    when(mockEmailService.sendEmail(any(String[].class), eq("content"))).thenReturn(false);

    // Run the test
    final MockHttpServletResponse response = mockMvc.perform(post("/sys/launchMessage")
                        .content("content").contentType(MediaType.APPLICATION_JSON)
                        .accept(MediaType.APPLICATION_JSON))
                .andReturn().getResponse();

    // Verify the results
    assertThat(response.getStatus()).isEqualTo(HttpStatus.OK.value());
    assertThat(response.getContentAsString()).isEqualTo("expectedResponse");
    verify(mockEmailService).sendEmail(any(String[].class), eq("content"));
}
```

6.1.2.2 addSyslog

（1）用例说明。其中参数均为 SyslogForm 的封装属性。

序号	输入参数	期望结果	参数说明
SS_SCON_6		code:500	传空
SS_SCON_7	{objtypeId:20000}	code:500	类别 id 非法
SS_SCON_8	{objtypeId:10000, objId:10000, operator: 10000, oper:10000}	code:200	正确用例
SS_SCON_9	{objtypeId:10000, objId:10000, oper:10000}	code:500	操作人为空
SS_SCON_10	{objtypeId:10000, operator: 10000, oper:10000}	code:500	对象 id 为空

（2） 测试代码部分实例

```
@Test
void testAddLog() throws Exception {
    // Setup
    when(mockSyslogService.addLog(new SyslogForm())).thenReturn(0L);

    // Run the test
    final MockHttpServletResponse response = mockMvc.perform(post("/sys/addLog")
                        .content("content").contentType(MediaType.APPLICATION_JSON)
                        .accept(MediaType.APPLICATION_JSON))
                .andReturn().getResponse();

    // Verify the results
    assertThat(response.getStatus()).isEqualTo(HttpStatus.OK.value());
```

```
    assertThat(response.getContentAsString()).isEqualTo("expectedResponse");
}
```

6.2 Service 层

6.2.1 测试描述

SystemService 模块的 Service 层单元测试主要是通过传入测试人员设置好的参数，查看 Service 层各个函数的具体输出是否符合逻辑，比如返回的数量，返回的属性，是否会抛出异常等等。涉及到的 Service 类包括 SyslogService，MessageService。

（1）测试对象

Service 层的测试对象即为各个 Service 类中的具体方法，详细描述如下表所示：

类名	MessageService	
方法名	方法描述	参数说明
launchMessage	发送信息	传入信息基本内容
类名	SyslogService	
方法名	方法描述	参数说明
addLog	添加日志	传入日志内容

（2）测试用例设计

Service 层的职责在于处理具体的业务逻辑。其中 SystemService 模块中的 Service 层主要是为了后台管理服务的，主要业务逻辑为数据库的增删改查，而对于算法没有什么要求，所以对于该 Service 层的单元测试主要是为了校验其返回值的主要属性，包括数量、主键是否正确，至于是否符合业务，需要到接口测试才能体现。

6.2.2 测试用例

对于每个测试对象都给出多个测试用例，以覆盖所有的参数输入情况，包括正常输入、边界值输入、空指输入等传参输入，并给出期望输出，和实例代码。

6.2.2.1 launchMessage

（1）用例说明。其中参数均为 MessageForm 的封装属性。

序号	输入参数	期望结果	参数说明
SS_SSER_1		flag:false	传空
SS_SSER_2	scope:10000,specificUsers:[10000,10001],content:”xxx”	flag:true,messageId:非空	正确用例
SS_SSER_3	scope:10000,specificUsers:[10000,10001]	flag:false	内容为空

（2） 测试代码部分实例

```
@Test
void testLaunchMessage() {
    // Setup
    final MessageForm messageForm = new MessageForm();
    messageForm.setMessageId(0L);
    messageForm.setCreateTime(Date.valueOf(LocalDate.of(2020, 1, 1)));
    messageForm.setDeleteMark("deleteMark");
    messageForm.setScope(0);
    messageForm.setScopeDetail("scopeDetail");
    messageForm.setContent("content");
    messageForm.setSpecificUsers(Arrays.asList(0L));
    messageForm.setTitle("title");
    messageForm.setEmails(new String[]{"emails"});

    // Run the test
    final Map<String, Object> result = messageServiceImplUnderTest.launchMessage(messageForm);

    // Verify the results
}
```

6.2.2.2 addSyslog

（1）用例说明。其中参数均为 SyslogForm 的封装属性。

序号	输入参数	期望结果	参数说明
SS_SSER_4		flag:false	传空
SS_SSER_5	objtypeId:10000, objId:10000, operator: 10000, oper:10000	flag:true,syslogId:非空	正确用例
SS_SSER_6	objtypeId:10000, objId:10000, operator: 10000	flag:false	操作为空

（2） 测试代码部分实例

```
@Test
void testAddLog() {
    // Setup
    final SyslogForm syslogForm = new SyslogForm();
    syslogForm.setLogId(0L);
    syslogForm.setCreateTime(Date.valueOf(LocalDate.of(2020, 1, 1)));
    syslogForm.setDeleteMark("deleteMark");
    syslogForm.setObjtypeId(0);
    syslogForm.setObjId(0L);
    syslogForm.setOperator(0L);
    syslogForm.setOperation(0);
    syslogForm.setMessage("message");
```

```
    syslogForm.setObjtype("objtype");

    when(mockRedisTemplate.opsForValue()).thenReturn(null);

    // Run the test
    final Long result = syslogServiceImplUnderTest.addLog(syslogForm);

    // Verify the results
    assertThat(result).isEqualTo(0L);
}
```

七、HouseService 模块单元测试

7.1 Controller

7.1.1 测试描述

HouseService 模块的 Controller 层单元测试主要是通过模拟 Http 请求，对 PostController 类的各个方法处理参数的能力做测试。

（1）测试对象

Controller 层的测试对象即为各个 Controller 类中的具体方法，详细描述如下表所示：

类名	PostController	
方法名	方法描述	参数说明
launchMessage	发送信息	传入信息基本内容

（2）测试用例设计

Controller 层的职责为处理校验参数的合法性，合法参数可进行下一步操作，不合法参数便直接返回不合法信息，为了提高用户体验，对于输入的参数 Controller 层均能返回 HTTP200 的状态码，但非法信息会夹带自定义的非法状态码。所以测试用例的设计不能只顾于返回的 HTTP 状态码，还要具体校验返回的信息。

7.1.2 测试用例

对于每个测试对象都给出多个测试用例，以覆盖所有的参数输入情况，包括正常输入、边界值输入、空指输入等传参输入，并给出期望输出，和实例代码。

7.1.2.1 testTest

（1）用例说明。

序号	输入参数	期望结果	参数说明
SS_SCON_1		code:500	传空

SS_SCON_2	{scope:10000, content:”xxxx”}	code:500	接收者为空
SS_SCON_3	{content:”xxx”,specificUsers:[10000, 10001]}	code:500	范围为空
SS_SCON_4	{scope:10000,specificUsers:[10000,10001]}	code:500	内容为空
SS_SCON_5	{scope:10000,specificUsers:[10000,10001],content:”xxx”}	code:200	正确用例

（2） 测试代码部分实例

```
@Test
void testTest() throws Exception {
    // Setup
    // Run the test
    final MockHttpServletResponse response = mockMvc.perform(post("/house/test")
                    .accept(MediaType.APPLICATION_JSON))
            .andReturn().getResponse();

    // Verify the results
    assertThat(response.getStatus()).isEqualTo(HttpStatus.OK.value());
    assertThat(response.getContentAsString()).isEqualTo("expectedResponse");
}
```

7.1.2.2 testPostHouse1

（1）用例说明。

序号	输入参数	期望结果	参数说明
SS_SCON_6		code:500	传空
SS_SCON_7	{objtypeId:20000}	code:500	类别 id 非法
SS_SCON_8	{objtypeId:10000, objId:10000, operator: 10000, oper:10000}	code:200	正确用例
SS_SCON_9	{objtypeId:10000, objId:10000, oper:10000}	code:500	操作人为空
SS_SCON_10	{objtypeId:10000, operator: 10000, oper:10000}	code:500	对象 id 为空

（2） 测试代码部分实例

```
@Test
void testPostHouse1() throws Exception {
    // Setup
    when(mockPostHouseService.addpostHouseInfo(new HouseinfoVo())).thenReturn(new HashMap<>());

    // Run the test
    final MockHttpServletResponse response = mockMvc.perform(post("/house/postinfo")
```

```
                .content("content").contentType(MediaType.APPLICATION_JSON)
                .accept(MediaType.APPLICATION_JSON))
        .andReturn().getResponse();

    // Verify the results
    assertThat(response.getStatus()).isEqualTo(HttpStatus.OK.value());
    assertThat(response.getContentAsString()).isEqualTo("expectedResponse");
}
@Test
void testPostHouse2() throws Exception {
    // Setup
    when(mockPostHouseService.addHouse(new HouseinfoVo(), Arrays.asList(new TagVo()),
            Arrays.asList(new ContactVo()),
            Arrays.asList())).thenReturn(new HashMap<>());

    // Run the test
    final MockHttpServletResponse response = mockMvc.perform(multipart("/house/postHouse")
                .file(new MockMultipartFile("houseinfoVo", "originalFilename",
MediaType.APPLICATION_JSON_VALUE,
                        "content".getBytes()))
                .file(new MockMultipartFile("fileinfo", "originalFilename",
MediaType.APPLICATION_JSON_VALUE,
                        "content".getBytes()))
                .file(new MockMultipartFile("tagList", "originalFilename",
MediaType.APPLICATION_JSON_VALUE,
                        "content".getBytes()))
                .file(new MockMultipartFile("contactList", "originalFilename",
MediaType.APPLICATION_JSON_VALUE,
                        "content".getBytes()))
                .accept(MediaType.APPLICATION_JSON))
        .andReturn().getResponse();

    // Verify the results
    assertThat(response.getStatus()).isEqualTo(HttpStatus.OK.value());
    assertThat(response.getContentAsString()).isEqualTo("expectedResponse");
}
```

7.1.2.3 testUploadHousePic

1）用例说明。

序号	输入参数	期望结果	参数说明

SS_SCON_6		code:500	传空
SS_SCON_7	{objtypeId:20000}	code:500	类别 id 非法
SS_SCON_8	{objtypeId:10000, objId:10000, operator: 10000, oper:10000}	code:200	正确用例
SS_SCON_9	{objtypeId:10000, objId:10000, oper:10000}	code:500	操作人为空
SS_SCON_10	{objtypeId:10000, operator: 10000, oper:10000}	code:500	对象 id 为空

（2） 测试代码部分实例

```
@Test
void testUploadHousePic() throws Exception {
    // Setup
    when(mockFilePicService.uploadHousePic(new FileName())).thenReturn(new HashMap<>());

    // Run the test
    final MockHttpServletResponse response = mockMvc.perform(multipart("/house/uploadHousePic")
                    .file(new MockMultipartFile("files", "originalFilename",
MediaType.APPLICATION_JSON_VALUE,
                                                "content".getBytes()))
                    .accept(MediaType.APPLICATION_JSON))
            .andReturn().getResponse();

    // Verify the results
    assertThat(response.getStatus()).isEqualTo(HttpStatus.OK.value());
    assertThat(response.getContentAsString()).isEqualTo("expectedResponse");
}
```

7.1.2.4 testUpdateHouse

1）用例说明。

序号	输入参数	期望结果	参数说明
SS_SCON_6		code:500	传空
SS_SCON_7	{objtypeId:20000}	code:500	类别 id 非法
SS_SCON_8	{objtypeId:10000, objId:10000, operator: 10000, oper:10000}	code:200	正确用例
SS_SCON_9	{objtypeId:10000, objId:10000, oper:10000}	code:500	操作人为空
SS_SCON_10	{objtypeId:10000, operator: 10000, oper:10000}	code:500	对象 id 为空

（3）测试代码部分实例

```
@Test
void testUpdateHouse() throws Exception {
    // Setup
    when(mockPostHouseService.updateHouseInfo(new HouseinfoVo())).thenReturn(new HashMap<>());

    // Run the test
    final MockHttpServletResponse response = mockMvc.perform(post("/house/updateinfo")
                        .content("content").contentType(MediaType.APPLICATION_JSON)
                        .accept(MediaType.APPLICATION_JSON))
            .andReturn().getResponse();

    // Verify the results
    assertThat(response.getStatus()).isEqualTo(HttpStatus.OK.value());
    assertThat(response.getContentAsString()).isEqualTo("expectedResponse");
}
```

7.1.2.5 testUploadHousePic

1）用例说明。

序号	输入参数	期望结果	参数说明
SS_SCON_6		code:500	传空
SS_SCON_7	{objtypeId:20000}	code:500	类别 id 非法
SS_SCON_8	{objtypeId:10000, objId:10000, operator: 10000, oper:10000}	code:200	正确用例
SS_SCON_9	{objtypeId:10000, objId:10000, oper:10000}	code:500	操作人为空
SS_SCON_10	{objtypeId:10000, operator: 10000, oper:10000}	code:500	对象 id 为空

（2） 测试代码部分实例

```
@Test
void testDeleteHouse() throws Exception {
    // Setup
    when(mockPostHouseService.deleteHouseInfo(0)).thenReturn(new HashMap<>());

    // Run the test
    final MockHttpServletResponse response = mockMvc.perform(post("/house/deleteinfo")
                        .content("content").contentType(MediaType.APPLICATION_JSON)
                        .accept(MediaType.APPLICATION_JSON))
            .andReturn().getResponse();
```

```
    // Verify the results
    assertThat(response.getStatus()).isEqualTo(HttpStatus.OK.value());
    assertThat(response.getContentAsString()).isEqualTo("expectedResponse");
    verify(mockAmqpTemplate).convertAndSend("queue", "o");
}
```

7.1.2.6 testDeleteHouse

1）用例说明。

序号	输入参数	期望结果	参数说明
SS_SCON_6		code:500	传空
SS_SCON_7	{objtypeId:20000}	code:500	类别 id 非法
SS_SCON_8	{objtypeId:10000, objId:10000, operator: 10000, oper:10000}	code:200	正确用例
SS_SCON_9	{objtypeId:10000, objId:10000, oper:10000}	code:500	操作人为空
SS_SCON_10	{objtypeId:10000, operator: 10000, oper:10000}	code:500	对象 id 为空

（3） 测试代码部分实例

```
@Test
void testDeleteHouse() throws Exception {
    // Setup
    when(mockPostHouseService.deleteHouseInfo(0)).thenReturn(new HashMap<>());

    // Run the test
    final MockHttpServletResponse response = mockMvc.perform(post("/house/deleteinfo")
                    .content("content").contentType(MediaType.APPLICATION_JSON)
                    .accept(MediaType.APPLICATION_JSON))
            .andReturn().getResponse();

    // Verify the results
    assertThat(response.getStatus()).isEqualTo(HttpStatus.OK.value());
    assertThat(response.getContentAsString()).isEqualTo("expectedResponse");
    verify(mockAmqpTemplate).convertAndSend("queue", "o");
}
```

7.1.2.7 testDeleteHouse_AmqpTemplateThrowsAmqpException

1）用例说明。

序号	输入参数	期望结果	参数说明
SS_SCON_6		code:500	传空
SS_SCON_7	{objtypeId:20000}	code:500	类别 id 非法
SS_SCON_8	{objtypeId:10000, objId:10000, operator: 10000, oper:10000}	code:200	正确用例
SS_SCON_9	{objtypeId:10000, objId:10000, oper:10000}	code:500	操作人为空
SS_SCON_10	{objtypeId:10000, operator: 10000, oper:10000}	code:500	对象 id 为空

（2） 测试代码部分实例

```
@Test
void testDeleteHouse_AmqpTemplateThrowsAmqpException() throws Exception {
    // Setup
    when(mockPostHouseService.deleteHouseInfo(0)).thenReturn(new HashMap<>());
    doThrow(AmqpException.class).when(mockAmqpTemplate).convertAndSend("queue", "o");

    // Run the test
    final MockHttpServletResponse response = mockMvc.perform(post("/house/deleteinfo")
                    .content("content").contentType(MediaType.APPLICATION_JSON)
                    .accept(MediaType.APPLICATION_JSON))
            .andReturn().getResponse();

    // Verify the results
    assertThat(response.getStatus()).isEqualTo(HttpStatus.OK.value());
    assertThat(response.getContentAsString()).isEqualTo("expectedResponse");
}
```

7.1.2.8 testHousedetial

1）用例说明。

序号	输入参数	期望结果	参数说明
SS_SCON_6		code:500	传空
SS_SCON_7	{objtypeId:20000}	code:500	类别 id 非法
SS_SCON_8	{objtypeId:10000, objId:10000, operator: 10000, oper:10000}	code:200	正确用例
SS_SCON_9	{objtypeId:10000, objId:10000, oper:10000}	code:500	操作人为空
SS_SCON_10	{objtypeId:10000, operator: 10000, oper:10000}	code:500	对象 id 为空

（3） 测试代码部分实例

```
@Test
void testHousedetial() throws Exception {
    // Setup
    when(mockPostHouseService.houseById(0)).thenReturn(new HashMap<>());

    // Run the test
    final MockHttpServletResponse response = mockMvc.perform(post("/house/housedetail")
                    .content("content").contentType(MediaType.APPLICATION_JSON)
                    .accept(MediaType.APPLICATION_JSON))
            .andReturn().getResponse();

    // Verify the results
    assertThat(response.getStatus()).isEqualTo(HttpStatus.OK.value());
    assertThat(response.getContentAsString()).isEqualTo("expectedResponse");
}
```

7.1.2.9 testSelectHouse

1）用例说明。

序号	输入参数	期望结果	参数说明
SS_SCON_6		code:500	传空
SS_SCON_7	{objtypeId:20000}	code:500	类别 id 非法
SS_SCON_8	{objtypeId:10000, objId:10000, operator: 10000, oper:10000}	code:200	正确用例
SS_SCON_9	{objtypeId:10000, objId:10000, oper:10000}	code:500	操作人为空
SS_SCON_10	{objtypeId:10000, operator: 10000, oper:10000}	code:500	对象 id 为空

（2） 测试代码部分实例

```
@Test
void testSelectHouse() throws Exception {
    // Setup
    when(mockPostHouseService.selcetHouse(new HashMap<>())).thenReturn(new HashMap<>());

    // Run the test
    final MockHttpServletResponse response = mockMvc.perform(post("/house/selectHouse")
                    .content("content").contentType(MediaType.APPLICATION_JSON)
```

```
                .accept(MediaType.APPLICATION_JSON))
        .andReturn().getResponse();

    // Verify the results
    assertThat(response.getStatus()).isEqualTo(HttpStatus.OK.value());
    assertThat(response.getContentAsString()).isEqualTo("expectedResponse");
}
```

7.1.2.10 testSelectHouseForAdmin

1）用例说明。

序号	输入参数	期望结果	参数说明
SS_SCON_6		code:500	传空
SS_SCON_7	{objtypeId:20000}	code:500	类别 id 非法
SS_SCON_8	{objtypeId:10000, objId:10000, operator: 10000, oper:10000}	code:200	正确用例
SS_SCON_9	{objtypeId:10000, objId:10000, oper:10000}	code:500	操作人为空
SS_SCON_10	{objtypeId:10000, operator: 10000, oper:10000}	code:500	对象 id 为空

（2） 测试代码部分实例

```
@Test
void testSelectHouseForAdmin() throws Exception {
    // Setup
    when(mockPostHouseService.selectBycondition(new HashMap<>())).thenReturn(new HashMap<>());

    // Run the test
    final MockHttpServletResponse response = mockMvc.perform(post("/house/selectHouseAdmin")
                .content("content").contentType(MediaType.APPLICATION_JSON)
                .accept(MediaType.APPLICATION_JSON))
        .andReturn().getResponse();

    // Verify the results
    assertThat(response.getStatus()).isEqualTo(HttpStatus.OK.value());
    assertThat(response.getContentAsString()).isEqualTo("expectedResponse");
}
```

7.1.2.11 testUserHouse

1）用例说明。

序号	输入参数	期望结果	参数说明
SS_SCON_6		code:500	传空
SS_SCON_7	{objtypeId:20000}	code:500	类别 id 非法
SS_SCON_8	{objtypeId:10000, objId:10000, operator: 10000, oper:10000}	code:200	正确用例
SS_SCON_9	{objtypeId:10000, objId:10000, oper:10000}	code:500	操作人为空
SS_SCON_10	{objtypeId:10000, operator: 10000, oper:10000}	code:500	对象 id 为空

（2） 测试代码部分实例

```
@Test
void testUserHouse() throws Exception {
    // Setup
    when(mockPostHouseService.selectBycondition(new HashMap<>())).thenReturn(new HashMap<>());

    // Run the test
    final MockHttpServletResponse response = mockMvc.perform(post("/house/userHouse")
                        .content("content").contentType(MediaType.APPLICATION_JSON)
                        .accept(MediaType.APPLICATION_JSON))
            .andReturn().getResponse();

    // Verify the results
    assertThat(response.getStatus()).isEqualTo(HttpStatus.OK.value());
    assertThat(response.getContentAsString()).isEqualTo("expectedResponse");
}
```

7.1.2.12 testPostconnect

1）用例说明。

序号	输入参数	期望结果	参数说明
SS_SCON_6		code:500	传空
SS_SCON_7	{objtypeId:20000}	code:500	类别 id 非法
SS_SCON_8	{objtypeId:10000, objId:10000, operator: 10000, oper:10000}	code:200	正确用例
SS_SCON_9	{objtypeId:10000, objId:10000, oper:10000}	code:500	操作人为空
SS_SCON_10	{objtypeId:10000, operator: 10000, oper:10000}	code:500	对象 id 为空

（2） 测试代码部分实例

```
@Test
void testPostconnect() throws Exception {
    // Setup
    when(mockPostContactService.addPostContact(Arrays.asList(new ContactVo()))).thenReturn(new HashMap<>());

    // Run the test
    final MockHttpServletResponse response = mockMvc.perform(post("/house/postcontact")
                        .content("content").contentType(MediaType.APPLICATION_JSON)
                        .accept(MediaType.APPLICATION_JSON))
                .andReturn().getResponse();

    // Verify the results
    assertThat(response.getStatus()).isEqualTo(HttpStatus.OK.value());
    assertThat(response.getContentAsString()).isEqualTo("expectedResponse");
}
```

7.1.2.13 testUpdateconnect

1）用例说明。

序号	输入参数	期望结果	参数说明
SS_SCON_6		code:500	传空
SS_SCON_7	{objtypeId:20000}	code:500	类别 id 非法
SS_SCON_8	{objtypeId:10000, objId:10000, operator: 10000, oper:10000}	code:200	正确用例
SS_SCON_9	{objtypeId:10000, objId:10000, oper:10000}	code:500	操作人为空
SS_SCON_10	{objtypeId:10000, operator: 10000, oper:10000}	code:500	对象 id 为空

（2） 测试代码部分实例

```
@Test
void testUpdateconnect() throws Exception {
    // Setup
    when(mockPostContactService.updateContact(new ContactVo())).thenReturn(new HashMap<>());

    // Run the test
    final MockHttpServletResponse response = mockMvc.perform(post("/house/updatecontact")
                        .content("content").contentType(MediaType.APPLICATION_JSON)
                        .accept(MediaType.APPLICATION_JSON))
```

```
                .andReturn().getResponse();

        // Verify the results
        assertThat(response.getStatus()).isEqualTo(HttpStatus.OK.value());
        assertThat(response.getContentAsString()).isEqualTo("expectedResponse");
}
```

7.1.2.14 testDeleteconnect

1）用例说明。

序号	输入参数	期望结果	参数说明
SS_SCON_6		code:500	传空
SS_SCON_7	{objtypeId:20000}	code:500	类别 id 非法
SS_SCON_8	{objtypeId:10000, objId:10000, operator: 10000, oper:10000}	code:200	正确用例
SS_SCON_9	{objtypeId:10000, objId:10000, oper:10000}	code:500	操作人为空
SS_SCON_10	{objtypeId:10000, operator: 10000, oper:10000}	code:500	对象 id 为空

（2） 测试代码部分实例

```
@Test
void testDeleteconnect() throws Exception {
        // Setup
        when(mockPostContactService.deleteContact(0)).thenReturn(new HashMap<>());

        // Run the test
        final MockHttpServletResponse response = mockMvc.perform(post("/house/deletecontact")
                        .content("content").contentType(MediaType.APPLICATION_JSON)
                        .accept(MediaType.APPLICATION_JSON))
                .andReturn().getResponse();

        // Verify the results
        assertThat(response.getStatus()).isEqualTo(HttpStatus.OK.value());
        assertThat(response.getContentAsString()).isEqualTo("expectedResponse");
}
```

7.1.2.15 testSelectContact

1）用例说明。

序号	输入参数	期望结果	参数说明
SS_SCON_6		code:500	传空
SS_SCON_7	{objtypeId:20000}	code:500	类别 id 非法
SS_SCON_8	{objtypeId:10000, objId:10000, operator: 10000, oper:10000}	code:200	正确用例
SS_SCON_9	{objtypeId:10000, objId:10000, oper:10000}	code:500	操作人为空
SS_SCON_10	{objtypeId:10000, operator: 10000, oper:10000}	code:500	对象 id 为空

（2） 测试代码部分实例

```
@Test
void testSelectContact() throws Exception {
    // Setup
    when(mockPostContactService.selectContact(new HashMap<>())).thenReturn(new HashMap<>());

    // Run the test
    final MockHttpServletResponse response = mockMvc.perform(post("/house/selectcontact")
                        .content("content").contentType(MediaType.APPLICATION_JSON)
                        .accept(MediaType.APPLICATION_JSON))
            .andReturn().getResponse();

    // Verify the results
    assertThat(response.getStatus()).isEqualTo(HttpStatus.OK.value());
    assertThat(response.getContentAsString()).isEqualTo("expectedResponse");
}
```

7.1.2.16 testSelectContactAll

1）用例说明。

序号	输入参数	期望结果	参数说明
SS_SCON_6		code:500	传空
SS_SCON_7	{objtypeId:20000}	code:500	类别 id 非法
SS_SCON_8	{objtypeId:10000, objId:10000, operator: 10000, oper:10000}	code:200	正确用例
SS_SCON_9	{objtypeId:10000, objId:10000, oper:10000}	code:500	操作人为空
SS_SCON_10	{objtypeId:10000, operator: 10000, oper:10000}	code:500	对象 id 为空

（2） 测试代码部分实例

```
@Test
void testSelectContactAll() throws Exception {
    // Setup
    when(mockPostContactService.selectContactAll(new HashMap<>())).thenReturn(new HashMap<>());

    // Run the test
    final MockHttpServletResponse response = mockMvc.perform(post("/house/selectcontactAdmin")
                        .content("content").contentType(MediaType.APPLICATION_JSON)
                        .accept(MediaType.APPLICATION_JSON))
            .andReturn().getResponse();

    // Verify the results
    assertThat(response.getStatus()).isEqualTo(HttpStatus.OK.value());
    assertThat(response.getContentAsString()).isEqualTo("expectedResponse");
}
```

7.1.2.17 testPostconnecttype

1）用例说明。

序号	输入参数	期望结果	参数说明
SS_SCON_6		code:500	传空
SS_SCON_7	{objtypeId:20000}	code:500	类别 id 非法
SS_SCON_8	{objtypeId:10000, objId:10000, operator: 10000, oper:10000}	code:200	正确用例
SS_SCON_9	{objtypeId:10000, objId:10000, oper:10000}	code:500	操作人为空
SS_SCON_10	{objtypeId:10000, operator: 10000, oper:10000}	code:500	对象 id 为空

（2） 测试代码部分实例

```
@Test
void testPostconnecttype() throws Exception {
    // Setup
    when(mockContactTypeService.addContactType(new ContactTypeVo())).thenReturn(new HashMap<>());

    // Run the test
    final MockHttpServletResponse response = mockMvc.perform(post("/house/postcontacttype")
                        .content("content").contentType(MediaType.APPLICATION_JSON)
                        .accept(MediaType.APPLICATION_JSON))
            .andReturn().getResponse();
```

```
    // Verify the results
    assertThat(response.getStatus()).isEqualTo(HttpStatus.OK.value());
    assertThat(response.getContentAsString()).isEqualTo("expectedResponse");
}
```

7.1.2.18 testUpdateconnecttype

1）用例说明。

序号	输入参数	期望结果	参数说明
SS_SCON_6		code:500	传空
SS_SCON_7	{objtypeId:20000}	code:500	类别 id 非法
SS_SCON_8	{objtypeId:10000, objId:10000, operator: 10000, oper:10000}	code:200	正确用例
SS_SCON_9	{objtypeId:10000, objId:10000, oper:10000}	code:500	操作人为空
SS_SCON_10	{objtypeId:10000, operator: 10000, oper:10000}	code:500	对象 id 为空

（2） 测试代码部分实例

```
@Test
void testUpdateconnecttype() throws Exception {
    // Setup
    when(mockContactTypeService.updateContactType(new ContactTypeVo())).thenReturn(new HashMap<>());

    // Run the test
    final MockHttpServletResponse response = mockMvc.perform(post("/house/updatecontacttype")
                    .content("content").contentType(MediaType.APPLICATION_JSON)
                    .accept(MediaType.APPLICATION_JSON))
            .andReturn().getResponse();

    // Verify the results
    assertThat(response.getStatus()).isEqualTo(HttpStatus.OK.value());
    assertThat(response.getContentAsString()).isEqualTo("expectedResponse");
}
```

7.1.2.19 testDeleteconnecttype

1）用例说明。

序号	输入参数	期望结果	参数说明

SS_SCON_6		code:500	传空
SS_SCON_7	{objtypeId:20000}	code:500	类别 id 非法
SS_SCON_8	{objtypeId:10000, objId:10000, operator: 10000, oper:10000}	code:200	正确用例
SS_SCON_9	{objtypeId:10000, objId:10000, oper:10000}	code:500	操作人为空
SS_SCON_10	{objtypeId:10000, operator: 10000, oper:10000}	code:500	对象 id 为空

（2） 测试代码部分实例

```
@Test
void testDeleteconnecttype() throws Exception {
    // Setup
    when(mockContactTypeService.deleteContactType(0)).thenReturn(new HashMap<>());

    // Run the test
    final MockHttpServletResponse response = mockMvc.perform(post("/house/deletecontacttype")
                    .content("content").contentType(MediaType.APPLICATION_JSON)
                    .accept(MediaType.APPLICATION_JSON))
            .andReturn().getResponse();

    // Verify the results
    assertThat(response.getStatus()).isEqualTo(HttpStatus.OK.value());
    assertThat(response.getContentAsString()).isEqualTo("expectedResponse");
}
```

7.1.2.20 testSelectContacttype

1）用例说明。

序号	输入参数	期望结果	参数说明
SS_SCON_6		code:500	传空
SS_SCON_7	{objtypeId:20000}	code:500	类别 id 非法
SS_SCON_8	{objtypeId:10000, objId:10000, operator: 10000, oper:10000}	code:200	正确用例
SS_SCON_9	{objtypeId:10000, objId:10000, oper:10000}	code:500	操作人为空
SS_SCON_10	{objtypeId:10000, operator: 10000, oper:10000}	code:500	对象 id 为空

（2） 测试代码部分实例

```
@Test
void testSelectContacttype() throws Exception {
    // Setup
    when(mockContactTypeService.selectContactType(new HashMap<>(), false)).thenReturn(new HashMap<>());

    // Run the test
    final MockHttpServletResponse response = mockMvc.perform(post("/house/selectcontacttype")
                        .content("content").contentType(MediaType.APPLICATION_JSON)
                        .accept(MediaType.APPLICATION_JSON))
            .andReturn().getResponse();

    // Verify the results
    assertThat(response.getStatus()).isEqualTo(HttpStatus.OK.value());
    assertThat(response.getContentAsString()).isEqualTo("expectedResponse");
}
```

7.1.2.21 testSelectContacttypeAdmin

1）用例说明。

序号	输入参数	期望结果	参数说明
SS_SCON_6		code:500	传空
SS_SCON_7	{objtypeId:20000}	code:500	类别 id 非法
SS_SCON_8	{objtypeId:10000, objId:10000, operator: 10000, oper:10000}	code:200	正确用例
SS_SCON_9	{objtypeId:10000, objId:10000, oper:10000}	code:500	操作人为空
SS_SCON_10	{objtypeId:10000, operator: 10000, oper:10000}	code:500	对象 id 为空

（2） 测试代码部分实例

```
@Test
void testSelectContacttypeAdmin() throws Exception {
    // Setup
    when(mockContactTypeService.selectContactType(new HashMap<>(), false)).thenReturn(new HashMap<>());

    // Run the test
    final MockHttpServletResponse response = mockMvc.perform(post("/house/selectcontacttypeAdmin")
                        .content("content").contentType(MediaType.APPLICATION_JSON)
                        .accept(MediaType.APPLICATION_JSON))
            .andReturn().getResponse();
```

```
    // Verify the results
    assertThat(response.getStatus()).isEqualTo(HttpStatus.OK.value());
    assertThat(response.getContentAsString()).isEqualTo("expectedResponse");
}
```

7.1.2.22 testPosttag

1）用例说明。

序号	输入参数	期望结果	参数说明
SS_SCON_6		code:500	传空
SS_SCON_7	{objtypeId:20000}	code:500	类别 id 非法
SS_SCON_8	{objtypeId:10000, objId:10000, operator: 10000, oper:10000}	code:200	正确用例
SS_SCON_9	{objtypeId:10000, objId:10000, oper:10000}	code:500	操作人为空
SS_SCON_10	{objtypeId:10000, operator: 10000, oper:10000}	code:500	对象 id 为空

（2） 测试代码部分实例

```
@Test
void testPosttag() throws Exception {
    // Setup
    when(mockPostTagService.addPostTag(Arrays.asList(new TagVo()))).thenReturn(new HashMap<>());

    // Run the test
    final MockHttpServletResponse response = mockMvc.perform(post("/house/posttag")
                    .content("content").contentType(MediaType.APPLICATION_JSON)
                    .accept(MediaType.APPLICATION_JSON))
            .andReturn().getResponse();

    // Verify the results
    assertThat(response.getStatus()).isEqualTo(HttpStatus.OK.value());
    assertThat(response.getContentAsString()).isEqualTo("expectedResponse");
}
```

7.1.2.23 testDeletetag

1）用例说明。

序号	输入参数	期望结果	参数说明

SS_SCON_6		code:500	传空
SS_SCON_7	{objtypeId:20000}	code:500	类别 id 非法
SS_SCON_8	{objtypeId:10000, objId:10000, operator: 10000, oper:10000}	code:200	正确用例
SS_SCON_9	{objtypeId:10000, objId:10000, oper:10000}	code:500	操作人为空
SS_SCON_10	{objtypeId:10000, operator: 10000, oper:10000}	code:500	对象 id 为空

（2） 测试代码部分实例

```
@Test
void testDeletetag() throws Exception {
    // Setup
    when(mockPostTagService.deleteTag(0)).thenReturn(new HashMap<>());

    // Run the test
    final MockHttpServletResponse response = mockMvc.perform(post("/house/deletetag")
                    .content("content").contentType(MediaType.APPLICATION_JSON)
                    .accept(MediaType.APPLICATION_JSON))
            .andReturn().getResponse();

    // Verify the results
    assertThat(response.getStatus()).isEqualTo(HttpStatus.OK.value());
    assertThat(response.getContentAsString()).isEqualTo("expectedResponse");
}
```

7.1.2.24 testSelectTag

1）用例说明。

序号	输入参数	期望结果	参数说明
SS_SCON_6		code:500	传空
SS_SCON_7	{objtypeId:20000}	code:500	类别 id 非法
SS_SCON_8	{objtypeId:10000, objId:10000, operator: 10000, oper:10000}	code:200	正确用例
SS_SCON_9	{objtypeId:10000, objId:10000, oper:10000}	code:500	操作人为空
SS_SCON_10	{objtypeId:10000, operator: 10000, oper:10000}	code:500	对象 id 为空

（2） 测试代码部分实例

```
@Test
void testSelectTag() throws Exception {
    // Setup
    when(mockPostTagService.selectTag(new HashMap<>())).thenReturn(new HashMap<>());

    // Run the test
    final MockHttpServletResponse response = mockMvc.perform(post("/house/selectTag")
                        .content("content").contentType(MediaType.APPLICATION_JSON)
                        .accept(MediaType.APPLICATION_JSON))
            .andReturn().getResponse();

    // Verify the results
    assertThat(response.getStatus()).isEqualTo(HttpStatus.OK.value());
    assertThat(response.getContentAsString()).isEqualTo("expectedResponse");
}
```

7.1.2.25 testPosttagtype

1）用例说明。

序号	输入参数	期望结果	参数说明
SS_SCON_6		code:500	传空
SS_SCON_7	{objtypeId:20000}	code:500	类别 id 非法
SS_SCON_8	{objtypeId:10000, objId:10000, operator: 10000, oper:10000}	code:200	正确用例
SS_SCON_9	{objtypeId:10000, objId:10000, oper:10000}	code:500	操作人为空
SS_SCON_10	{objtypeId:10000, operator: 10000, oper:10000}	code:500	对象 id 为空

（2） 测试代码部分实例

```
@Test
void testPosttagtype() throws Exception {
    // Setup
    when(mockTagTypeService.addTagType(Arrays.asList(new TagTypeVo()))).thenReturn(new HashMap<>());

    // Run the test
    final MockHttpServletResponse response = mockMvc.perform(post("/house/posttagtype")
                        .content("content").contentType(MediaType.APPLICATION_JSON)
                        .accept(MediaType.APPLICATION_JSON))
            .andReturn().getResponse();
```

```
    // Verify the results
    assertThat(response.getStatus()).isEqualTo(HttpStatus.OK.value());
    assertThat(response.getContentAsString()).isEqualTo("expectedResponse");
}
```

7.1.2.26 testDeletetagtype

1）用例说明。

序号	输入参数	期望结果	参数说明
SS_SCON_6		code:500	传空
SS_SCON_7	{objtypeId:20000}	code:500	类别 id 非法
SS_SCON_8	{objtypeId:10000, objId:10000, operator: 10000, oper:10000}	code:200	正确用例
SS_SCON_9	{objtypeId:10000, objId:10000, oper:10000}	code:500	操作人为空
SS_SCON_10	{objtypeId:10000, operator: 10000, oper:10000}	code:500	对象 id 为空

（2） 测试代码部分实例

```
@Test
void testDeletetagtype() throws Exception {
    // Setup
    when(mockTagTypeService.deleteTagType(0)).thenReturn(new HashMap<>());

    // Run the test
    final MockHttpServletResponse response = mockMvc.perform(post("/house/deletetagtype")
                    .content("content").contentType(MediaType.APPLICATION_JSON)
                    .accept(MediaType.APPLICATION_JSON))
            .andReturn().getResponse();

    // Verify the results
    assertThat(response.getStatus()).isEqualTo(HttpStatus.OK.value());
    assertThat(response.getContentAsString()).isEqualTo("expectedResponse");
}
```

7.1.2.27 testSelecttagtype

1）用例说明。

序号	输入参数	期望结果	参数说明

SS_SCON_6		code:500	传空
SS_SCON_7	{objtypeId:20000}	code:500	类别 id 非法
SS_SCON_8	{objtypeId:10000, objId:10000, operator: 10000, oper:10000}	code:200	正确用例
SS_SCON_9	{objtypeId:10000, objId:10000, oper:10000}	code:500	操作人为空
SS_SCON_10	{objtypeId:10000, operator: 10000, oper:10000}	code:500	对象 id 为空

（2） 测试代码部分实例

```
@Test
void testSelecttagtype() throws Exception {
    // Setup
    when(mockTagTypeService.selectTagType(new HashMap<>())).thenReturn(new HashMap<>());

    // Run the test
    final MockHttpServletResponse response = mockMvc.perform(post("/house/selecttagtype")
                    .content("content").contentType(MediaType.APPLICATION_JSON)
                    .accept(MediaType.APPLICATION_JSON))
            .andReturn().getResponse();

    // Verify the results
    assertThat(response.getStatus()).isEqualTo(HttpStatus.OK.value());
    assertThat(response.getContentAsString()).isEqualTo("expectedResponse");
}
```

7.1.2.28 testReport

1）用例说明。

序号	输入参数	期望结果	参数说明
SS_SCON_6		code:500	传空
SS_SCON_7	{objtypeId:20000}	code:500	类别 id 非法
SS_SCON_8	{objtypeId:10000, objId:10000, operator: 10000, oper:10000}	code:200	正确用例
SS_SCON_9	{objtypeId:10000, objId:10000, oper:10000}	code:500	操作人为空
SS_SCON_10	{objtypeId:10000, operator: 10000, oper:10000}	code:500	对象 id 为空

（2） 测试代码部分实例

```
@Test
void testReport() throws Exception {
    // Setup
    when(mockReportClient.addReport(new HashMap<>())).thenReturn(new Result<>(new HashMap<>()));

    // Run the test
    final MockHttpServletResponse response = mockMvc.perform(post("/house/report")
                        .content("content").contentType(MediaType.APPLICATION_JSON)
                        .accept(MediaType.APPLICATION_JSON))
            .andReturn().getResponse();

    // Verify the results
    assertThat(response.getStatus()).isEqualTo(HttpStatus.OK.value());
    assertThat(response.getContentAsString()).isEqualTo("expectedResponse");
}
```

7.2 Service 层

7.2.1 测试描述

houseservice 模块的 Service 层单元测试主要是通过传入测试人员设置好的参数，查看 Service 层各个函数的具体输出是否符合逻辑，比如返回的数量，返回的属性，是否会抛出异常等等。涉及到的 Service 类包括 ContactTypeService，FilePicService,PostContactService,PostHouseServiceImpl,PostTagServiceImplTagTypeServiceImpl。

（1）测试对象

Service 层的测试对象即为各个 Service 类中的具体方法，包含 ContactTypeService，FilePicService,PostContactService,PostHouseServiceImpl,PostTagServiceImplTagTypeServiceImpl。

（2）测试用例设计

Service 层的职责在于处理具体的业务逻辑。其中 SystemService 模块中的 Service 层主要是为了后台管理服务的，主要业务逻辑为数据库的增删改查，而对于算法没有什么要求，所以对于该 Service 层的单元测试主要是为了校验其返回值的主要属性，包括数量、主键是否正确，至于是否符合业务，需要到接口测试才能体现。

7.2.2 测试用例

对于每个测试对象都给出多个测试用例，以覆盖所有的参数输入情况，包括正常输入、边界值输入、空指输入等传参输入，并给出期望输出，和实例代码。

7.2.2.1 testAddContactType

（1）用例说明。其中参数均为 MessageForm 的封装属性。

序号	输入参数	期望结果	参数说明
SS_SSER_1		flag:false	传空
SS_SSER_2	scope:10000,specificUsers:[10000,10001],content:”xxx”	flag:true,messageId:非空	正确用例
SS_SSER_3	scope:10000,specificUsers:[10000,10001]	flag:false	内容为空

（3） 测试代码部分实例

```
@Test
void testAddContactType() {
    // Setup
    final ContactTypeVo contactTypeVo = new ContactTypeVo();
    contactTypeVo.setTypeId(0);
    contactTypeVo.setCreateTime(Date.valueOf(LocalDate.of(2020, 1, 1)));
    contactTypeVo.setDeleteMark("deleteMark");
    contactTypeVo.setContactName("contactName");

    // Run the test
    final Map<String, Object> result = contactTypeServiceImplUnderTest.addContactType(contactTypeVo);

    // Verify the results
}
```

7.2.2.2 testUpdateContactType

（1）用例说明。其中参数均为 MessageForm 的封装属性。

序号	输入参数	期望结果	参数说明
SS_SSER_1		flag:false	传空
SS_SSER_2	scope:10000,specificUsers:[10000,10001],content:”xxx”	flag:true,messageId:非空	正确用例
SS_SSER_3	scope:10000,specificUsers:[10000,10001]	flag:false	内容为空

（2） 测试代码部分实例

```
@Test
void testUpdateContactType() {
    // Setup
```

```
    final ContactTypeVo contactTypeVo = new ContactTypeVo();
    contactTypeVo.setTypeId(0);
    contactTypeVo.setCreateTime(Date.valueOf(LocalDate.of(2020, 1, 1)));
    contactTypeVo.setDeleteMark("deleteMark");
    contactTypeVo.setContactName("contactName");

    // Run the test
    final Map<String, Object> result = contactTypeServiceImplUnderTest.updateContactType(contactTypeVo);

    // Verify the results
}
```

7.2.2.3 testDeleteContactType

（1）用例说明。其中参数均为 MessageForm 的封装属性。

序号	输入参数	期望结果	参数说明
SS_SSER_1		flag:false	传空
SS_SSER_2	scope:10000,specificUsers:[10000,10001],content:”xxx”	flag:true,messageId:非空	正确用例
SS_SSER_3	scope:10000,specificUsers:[10000,10001]	flag:false	内容为空

（2） 测试代码部分实例

```
@Test
void testDeleteContactType() {
    // Setup
    // Run the test
    final Map<String, Object> result = contactTypeServiceImplUnderTest.deleteContactType(0);

    // Verify the results
}
```

7.2.2.4 testSelectContactType

（1）用例说明。其中参数均为 MessageForm 的封装属性。

序号	输入参数	期望结果	参数说明
SS_SSER_1		flag:false	传空
SS_SSER_2	scope:10000,specificUsers:[10000,10001],content:”xxx”	flag:true,messageId:非空	正确用例

SS_SSER_3	scope:10000,specificUsers:[10000,10001]	flag:false	内容为空

（2） 测试代码部分实例

```
@Test
void testSelectContactType() {
    // Setup
    final Map<String, Object> selectCondition = new HashMap<>();

    // Run the test
    final Map<String, Object> result = contactTypeServiceImplUnderTest.selectContactType(selectCondition, false);

    // Verify the results
}
```

7.2.2.5 testTypeNameByid

（1）用例说明。其中参数均为 MessageForm 的封装属性。

序号	输入参数	期望结果	参数说明
SS_SSER_1		flag:false	传空
SS_SSER_2	scope:10000,specificUsers:[10000,10001],content:”xxx”	flag:true,messageId:非空	正确用例
SS_SSER_3	scope:10000,specificUsers:[10000,10001]	flag:false	内容为空

（2） 测试代码部分实例

```
@Test
void testTypeNameByid() {
    // Setup
    // Run the test
    final String result = contactTypeServiceImplUnderTest.TypeNameByid(0);

    // Verify the results
    assertThat(result).isEqualTo("contactName");
}
```

7.2.2.6 testUploadHousePic

（1）用例说明。其中参数均为 MessageForm 的封装属性。

序号	输入参数	期望结果	参数说明
SS_SSER_1		flag:false	传空
SS_SSER_2	scope:10000,specificUsers:[10000,10001],content:”xxx”	flag:true,messageId:非空	正确用例
SS_SSER_3	scope:10000,specificUsers:[10000,10001]	flag:false	内容为空

（2） 测试代码部分实例

```
@Test
void testUploadHousePic() {
    // Setup
    final FileName fileName = new FileName();
    fileName.setPicId(0);
    fileName.setCreateTime(Date.valueOf(LocalDate.of(2020, 1, 1)));
    fileName.setDeleteMark("deleteMark");
    fileName.setFilePath("filePath");
    fileName.setHouseId(0);

    // Run the test
    final Map<String, Object> result = filePicServiceImplUnderTest.uploadHousePic(fileName);

    // Verify the results
}
```

7.2.2.7 testReHousePic

（1）用例说明。其中参数均为 MessageForm 的封装属性。

序号	输入参数	期望结果	参数说明
SS_SSER_1		flag:false	传空
SS_SSER_2	scope:10000,specificUsers:[10000,10001],content:”xxx”	flag:true,messageId:非空	正确用例
SS_SSER_3	scope:10000,specificUsers:[10000,10001]	flag:false	内容为空

（2） 测试代码部分实例

```
@Test
void testReHousePic() {
```

```
    // Setup
    final FileName fileName = new FileName();
    fileName.setPicId(0);
    fileName.setCreateTime(Date.valueOf(LocalDate.of(2020, 1, 1)));
    fileName.setDeleteMark("deleteMark");
    fileName.setFilePath("filePath");
    fileName.setHouseId(0);

    // Run the test
    final Map<String, Object> result = filePicServiceImplUnderTest.reHousePic(fileName);

    // Verify the results
}
```

7.2.2.8 testAddPostContact

（1）用例说明。其中参数均为 MessageForm 的封装属性。

序号	输入参数	期望结果	参数说明
SS_SSER_1		flag:false	传空
SS_SSER_2	scope:10000,specificUsers:[10000,10001],content:”xxx”	flag:true,messageId:非空	正确用例
SS_SSER_3	scope:10000,specificUsers:[10000,10001]	flag:false	内容为空

（2） 测试代码部分实例

```
@Test
void testAddPostContact() {
    // Setup
    final ContactVo contactVo = new ContactVo();
    contactVo.setContactId(0);
    contactVo.setHouseId(0);
    contactVo.setContent("content");
    contactVo.setDeleteMark("deleteMark");
    contactVo.setTypeId(0);
    contactVo.setContactName("ContactName");
    final List<ContactVo> listContact = Arrays.asList(contactVo);

    // Run the test
    final Map<String, Object> result = postContactServiceImplUnderTest.addPostContact(listContact);
```

```
    // Verify the results
}
```

7.2.2.9 testUpdateContact

（1）用例说明。其中参数均为 MessageForm 的封装属性。

序号	输入参数	期望结果	参数说明
SS_SSER_1		flag:false	传空
SS_SSER_2	scope:10000,specificUsers:[10000,10001],content:”xxx”	flag:true,messageId:非空	正确用例
SS_SSER_3	scope:10000,specificUsers:[10000,10001]	flag:false	内容为空

（2） 测试代码部分实例

```
@Test
void testUpdateContact() {
    // Setup
    final ContactVo contactVo = new ContactVo();
    contactVo.setContactId(0);
    contactVo.setHouseId(0);
    contactVo.setContent("content");
    contactVo.setDeleteMark("deleteMark");
    contactVo.setTypeId(0);
    contactVo.setContactName("ContactName");

    // Run the test
    final Map<String, Object> result = postContactServiceImplUnderTest.updateContact(contactVo);

    // Verify the results
}
```

7.2.2.10 testDeleteContact

（1）用例说明。其中参数均为 MessageForm 的封装属性。

序号	输入参数	期望结果	参数说明
SS_SSER_1		flag:false	传空
SS_SSER_2	scope:10000,specificUsers:[10000,10001],content:”xxx”	flag:true,messageId:非空	正确用例

SS_SSER_3	scope:10000,specificUsers:[10000,10001]	flag:false	内容为空

（2） 测试代码部分实例

```
@Test
void testDeleteContact() {
    // Setup
    // Run the test
    final Map<String, Object> result = postContactServiceImplUnderTest.deleteContact(0);

    // Verify the results
}
```

7.2.2.11 testSelectContact

（1）用例说明。其中参数均为 MessageForm 的封装属性。

序号	输入参数	期望结果	参数说明
SS_SSER_1		flag:false	传空
SS_SSER_2	scope:10000,specificUsers:[10000,10001],content:”xxx”	flag:true,messageId:非空	正确用例
SS_SSER_3	scope:10000,specificUsers:[10000,10001]	flag:false	内容为空

（2） 测试代码部分实例

```
@Test
void testSelectContact() {
    // Setup
    final Map<String, Object> selectCondition = new HashMap<>();

    // Run the test
    final Map<String, Object> result = postContactServiceImplUnderTest.selectContact(selectCondition);

    // Verify the results
}
```

7.2.2.12 testSelectContactAll

（1）用例说明。其中参数均为 MessageForm 的封装属性。

序号	输入参数	期望结果	参数说明
SS_SSER_1		flag:false	传空
SS_SSER_2	scope:10000,specificUsers:[10000,10001],content:”xxx”	flag:true,messageId:非空	正确用例
SS_SSER_3	scope:10000,specificUsers:[10000,10001]	flag:false	内容为空

（2） 测试代码部分实例

```
@Test
void testSelectContactAll() {
    // Setup
    final Map<String, Object> selectCondition = new HashMap<>();

    // Run the test
    final Map<String, Object> result = postContactServiceImplUnderTest.selectContactAll(selectCondition);

    // Verify the results
}
```

7.2.2.13 testAddpostHouseInfo

（1）用例说明。其中参数均为 MessageForm 的封装属性。

序号	输入参数	期望结果	参数说明
SS_SSER_1		flag:false	传空
SS_SSER_2	scope:10000,specificUsers:[10000,10001],content:”xxx”	flag:true,messageId:非空	正确用例
SS_SSER_3	scope:10000,specificUsers:[10000,10001]	flag:false	内容为空

（2） 测试代码部分实例

```
@Test
void testAddpostHouseInfo() {
    // Setup
    final HouseinfoVo houseinfoVo = new HouseinfoVo();
    houseinfoVo.setUserId(0);
    houseinfoVo.setHouseId(0);
    houseinfoVo.setCountry("country");
    houseinfoVo.setProvince("province");
    houseinfoVo.setDeleteMark("deleteMark");
    houseinfoVo.setCity("city");
```

```
        houseinfoVo.setAddress("address");
        houseinfoVo.setGuests(0);
        houseinfoVo.setPets("pets");
        houseinfoVo.setDuration(0);
        houseinfoVo.setDescription("description");
        houseinfoVo.setTitle("title");
        houseinfoVo.setActive("active");
        houseinfoVo.setPageSize(0);
        houseinfoVo.setFileNames(Arrays.asList("value"));

        // Run the test
        final Map<String, Object> result = postHouseServiceImplUnderTest.addpostHouseInfo(houseinfoVo);

        // Verify the results
    }
```

7.2.2.14 testUpdateHouseInfo

（1）用例说明。其中参数均为 MessageForm 的封装属性。

序号	输入参数	期望结果	参数说明
SS_SSER_1		flag:false	传空
SS_SSER_2	scope:10000,specificUsers:[10000,10001],content:”xxx”	flag:true,messageId:非空	正确用例
SS_SSER_3	scope:10000,specificUsers:[10000,10001]	flag:false	内容为空

（2） 测试代码部分实例

```
@Test
void testUpdateHouseInfo() {
        // Setup
        final HouseinfoVo houseinfoVo = new HouseinfoVo();
        houseinfoVo.setUserId(0);
        houseinfoVo.setHouseId(0);
        houseinfoVo.setCountry("country");
        houseinfoVo.setProvince("province");
        houseinfoVo.setDeleteMark("deleteMark");
        houseinfoVo.setCity("city");
        houseinfoVo.setAddress("address");
        houseinfoVo.setGuests(0);
        houseinfoVo.setPets("pets");
```

```
    houseinfoVo.setDuration(0);
    houseinfoVo.setDescription("description");
    houseinfoVo.setTitle("title");
    houseinfoVo.setActive("active");
    houseinfoVo.setPageSize(0);
    houseinfoVo.setFileNames(Arrays.asList("value"));

    // Run the test
    final Map<String, Object> result = postHouseServiceImplUnderTest.updateHouseInfo(houseinfoVo);

    // Verify the results
}
```

7.2.2.15 testDeleteHouseInfo

（1）用例说明。其中参数均为 MessageForm 的封装属性。

序号	输入参数	期望结果	参数说明
SS_SSER_1		flag:false	传空
SS_SSER_2	scope:10000,specificUsers:[10000,10001],content:”xxx”	flag:true,messageId:非空	正确用例
SS_SSER_3	scope:10000,specificUsers:[10000,10001]	flag:false	内容为空

（2） 测试代码部分实例

```
@Test
void testDeleteHouseInfo() {
    // Setup
    // Run the test
    final Map<String, Object> result = postHouseServiceImplUnderTest.deleteHouseInfo(0);

    // Verify the results
}
```

7.2.2.16 testSelcetHouse

（1）用例说明。其中参数均为 MessageForm 的封装属性。

序号	输入参数	期望结果	参数说明
SS_SSER_1		flag:false	传空

SS_SSER_2	scope:10000,specificUsers:[10000,10001],content:”xxx”	flag:true,messageId:非空	正确用例
SS_SSER_3	scope:10000,specificUsers:[10000,10001]	flag:false	内容为空

（2） 测试代码部分实例

```
@Test
void testSelcetHouse() {
    // Setup
    final Map<String, Object> selectCondition = new HashMap<>();

    // Run the test
    final Map<String, Object> result = postHouseServiceImplUnderTest.selcetHouse(selectCondition);

    // Verify the results
}
```

7.2.2.17 testHouseById

（1）用例说明。其中参数均为 MessageForm 的封装属性。

序号	输入参数	期望结果	参数说明
SS_SSER_1		flag:false	传空
SS_SSER_2	scope:10000,specificUsers:[10000,10001],content:”xxx”	flag:true,messageId:非空	正确用例
SS_SSER_3	scope:10000,specificUsers:[10000,10001]	flag:false	内容为空

（2） 测试代码部分实例

```
@Test
void testHouseById() {
    // Setup
    when(mockPostTagService.selectTag(new HashMap<>())).thenReturn(new HashMap<>());
    when(mockTagTypeService.tagNameById(0)).thenReturn("tagName");
    when(mockPostContactService.selectContact(new HashMap<>())).thenReturn(new HashMap<>());
    when(mockContactTypeService.TypeNameByid(0)).thenReturn("ContactName");
    when(mockFilePicService.reHousePic(new FileName())).thenReturn(new HashMap<>());

    // Run the test
    final Map<String, Object> result = postHouseServiceImplUnderTest.houseById(0);
```

```
    // Verify the results
}
```

7.2.2.18 testSelectBycondition

（1）用例说明。其中参数均为 MessageForm 的封装属性。

序号	输入参数	期望结果	参数说明
SS_SSER_1		flag:false	传空
SS_SSER_2	scope:10000,specificUsers:[10000,10001],content:”xxx”	flag:true,messageId:非空	正确用例
SS_SSER_3	scope:10000,specificUsers:[10000,10001]	flag:false	内容为空

（2） 测试代码部分实例

```
@Test
void testSelectBycondition() {
    // Setup
    final Map<String, Object> selectCondition = new HashMap<>();
    when(mockFilePicService.reHousePic(new FileName())).thenReturn(new HashMap<>());

    // Run the test
    final Map<String, Object> result = postHouseServiceImplUnderTest.selectBycondition(selectCondition);

    // Verify the results
}
```

7.2.2.19 testAddHouse

（1）用例说明。其中参数均为 MessageForm 的封装属性。

序号	输入参数	期望结果	参数说明
SS_SSER_1		flag:false	传空
SS_SSER_2	scope:10000,specificUsers:[10000,10001],content:”xxx”	flag:true,messageId:非空	正确用例
SS_SSER_3	scope:10000,specificUsers:[10000,10001]	flag:false	内容为空

（2） 测试代码部分实例

```
@Test
void testAddHouse() {
        // Setup
        final HouseinfoVo houseinfoVo = new HouseinfoVo();
        houseinfoVo.setUserId(0);
        houseinfoVo.setHouseId(0);
        houseinfoVo.setCountry("country");
        houseinfoVo.setProvince("province");
        houseinfoVo.setDeleteMark("deleteMark");
        houseinfoVo.setCity("city");
        houseinfoVo.setAddress("address");
        houseinfoVo.setGuests(0);
        houseinfoVo.setPets("pets");
        houseinfoVo.setDuration(0);
        houseinfoVo.setDescription("description");
        houseinfoVo.setTitle("title");
        houseinfoVo.setActive("active");
        houseinfoVo.setPageSize(0);
        houseinfoVo.setFileNames(Arrays.asList("value"));

        final TagVo tagVo = new TagVo();
        tagVo.setTagId(0);
        tagVo.setHouseId(0);
        tagVo.setDeleteMark("deleteMark");
        tagVo.setTypeId(0);
        tagVo.setTagName("tagName");
        final List<TagVo> tagVoList = Arrays.asList(tagVo);
        final ContactVo contactVo = new ContactVo();
        contactVo.setContactId(0);
        contactVo.setHouseId(0);
        contactVo.setContent("content");
        contactVo.setDeleteMark("deleteMark");
        contactVo.setTypeId(0);
        contactVo.setContactName("ContactName");
        final List<ContactVo> contactVoList = Arrays.asList(contactVo);
        final List<MultipartFile> multipartFiles = Arrays.asList();
        when(mockPostTagService.addPostTag(Arrays.asList(new TagVo()))).thenReturn(new HashMap<>());
        when(mockPostContactService.addPostContact(Arrays.asList(new ContactVo()))).thenReturn(new HashMap<>());
        when(mockFilePicService.uploadHousePic(new FileName())).thenReturn(new HashMap<>());

        // Run the test
        final Map<String, Object> result = postHouseServiceImplUnderTest.addHouse(houseinfoVo, tagVoList, contactVoList,
                multipartFiles);
```

```
    // Verify the results
    verify(mockFilePicService).uploadHousePic(new FileName());
}
```

7.2.2.20 testAddPostTag

（1）用例说明。其中参数均为 MessageForm 的封装属性。

序号	输入参数	期望结果	参数说明
SS_SSER_1		flag:false	传空
SS_SSER_2	scope:10000,specificUsers:[10000,10001],content:”xxx”	flag:true,messageId:非空	正确用例
SS_SSER_3	scope:10000,specificUsers:[10000,10001]	flag:false	内容为空

（2） 测试代码部分实例

```
@Test
void testAddPostTag() {
    // Setup
    final TagVo tagVo = new TagVo();
    tagVo.setTagId(0);
    tagVo.setHouseId(0);
    tagVo.setDeleteMark("deleteMark");
    tagVo.setTypeId(0);
    tagVo.setTagName("tagName");
    final List<TagVo> list = Arrays.asList(tagVo);

    // Run the test
    final Map<String, Object> result = postTagServiceImplUnderTest.addPostTag(list);

    // Verify the results
}
```

7.2.2.21 testDeleteTag

（1）用例说明。其中参数均为 MessageForm 的封装属性。

序号	输入参数	期望结果	参数说明
SS_SSER_1		flag:false	传空
SS_SSER_2	scope:10000,specificUsers:[10000,10001],content:”xxx”	flag:true,messageId:非空	正确用例

SS_SSER_3	scope:10000,specificUsers:[10000,10001]	flag:false	内容为空

（2） 测试代码部分实例

```
@Test
void testDeleteTag() {
    // Setup
    // Run the test
    final Map<String, Object> result = postTagServiceImplUnderTest.deleteTag(0);

    // Verify the results
}
```

7.2.2.22 testSelectTag

（1）用例说明。其中参数均为 MessageForm 的封装属性。

序号	输入参数	期望结果	参数说明
SS_SSER_1		flag:false	传空
SS_SSER_2	scope:10000,specificUsers:[10000,10001],content:”xxx”	flag:true,messageId:非空	正确用例
SS_SSER_3	scope:10000,specificUsers:[10000,10001]	flag:false	内容为空

（2） 测试代码部分实例

```
@Test
void testSelectTag() {
    // Setup
    final Map<String, Object> selectCondition = new HashMap<>();

    // Run the test
    final Map<String, Object> result = postTagServiceImplUnderTest.selectTag(selectCondition);

    // Verify the results
}
```

7.2.2.23 testAddTagType

（1）用例说明。其中参数均为 MessageForm 的封装属性。

序号	输入参数	期望结果	参数说明
SS_SSER_1		flag:false	传空
SS_SSER_2	scope:10000,specificUsers:[10000,10001],content:”xxx”	flag:true,messageId:非空	正确用例
SS_SSER_3	scope:10000,specificUsers:[10000,10001]	flag:false	内容为空

（2） 测试代码部分实例

```
@Test
void testAddTagType() {
    // Setup
    final TagTypeVo tagTypeVo = new TagTypeVo();
    tagTypeVo.setTypeId(0);
    tagTypeVo.setCreateTime(Date.valueOf(LocalDate.of(2020, 1, 1)));
    tagTypeVo.setDeleteMark("deleteMark");
    tagTypeVo.setTagName("tagName");
    final List<TagTypeVo> list = Arrays.asList(tagTypeVo);

    // Run the test
    final Map<String, Object> result = tagTypeServiceImplUnderTest.addTagType(list);

    // Verify the results
}
```

7.2.2.24 testDeleteTagType

（1）用例说明。其中参数均为 MessageForm 的封装属性。

序号	输入参数	期望结果	参数说明
SS_SSER_1		flag:false	传空
SS_SSER_2	scope:10000,specificUsers:[10000,10001],content:”xxx”	flag:true,messageId:非空	正确用例
SS_SSER_3	scope:10000,specificUsers:[10000,10001]	flag:false	内容为空

（2） 测试代码部分实例

```
@Test
void testDeleteTagType() {
    // Setup
    // Run the test
    final Map<String, Object> result = tagTypeServiceImplUnderTest.deleteTagType(0);
```

```
    // Verify the results
}
```

7.2.2.25 testSelectTagType

（1）用例说明。其中参数均为 MessageForm 的封装属性。

序号	输入参数	期望结果	参数说明
SS_SSER_1		flag:false	传空
SS_SSER_2	scope:10000,specificUsers:[10000,10001],content:”xxx”	flag:true,messageId:非空	正确用例
SS_SSER_3	scope:10000,specificUsers:[10000,10001]	flag:false	内容为空

（2） 测试代码部分实例

```
@Test
void testSelectTagType() {
    // Setup
    final Map<String, Object> selectCondition = new HashMap<>();

    // Run the test
    final Map<String, Object> result = tagTypeServiceImplUnderTest.selectTagType(selectCondition);

    // Verify the results
}
```

7.2.2.26 testTagNameById

（1）用例说明。其中参数均为 MessageForm 的封装属性。

序号	输入参数	期望结果	参数说明
SS_SSER_1		flag:false	传空
SS_SSER_2	scope:10000,specificUsers:[10000,10001],content:”xxx”	flag:true,messageId:非空	正确用例
SS_SSER_3	scope:10000,specificUsers:[10000,10001]	flag:false	内容为空

（2） 测试代码部分实例

```
@Test
void testTagNameById() {
    // Setup
    // Run the test
    final String result = tagTypeServiceImplUnderTest.tagNameById(0);

    // Verify the results
    assertThat(result).isEqualTo("tagName");
}
```

八、SystemService 模块单元测试

8.1 Controller

8.1.1 测试描述

NewsServicet 模块的 Controller 层单元测试主要是通过模拟 Http 请求，对 NewsServicet、类的各个方法处理参数的能力做测试。

（1）测试对象

Controller 层的测试对象即为各个 Controller 类中的具体方法，

（2）测试用例设计

Controller 层的职责为处理校验参数的合法性，合法参数可进行下一步操作，不合法参数便直接返回不合法信息，为了提高用户体验，对于输入的参数 Controller 层均能返回 HTTP200 的状态码，但非法信息会夹带自定义的非法状态码。所以测试用例的设计不能只顾于返回的 HTTP 状态码，还要具体校验返回的信息。

8.1.2 测试用例

对于每个测试对象都给出多个测试用例，以覆盖所有的参数输入情况，包括正常输入、边界值输入、空指输入等传参输入，并给出期望输出，和实例代码。

8.1.2.1 testGet

（1）用例说明。其中参数均为 MessageForm 的封装属性。

序号	输入参数	期望结果	参数说明
SS_SCON_1		code:500	传空
SS_SCON_2	{scope:10000, content:”xxxx”}	code:500	接收者为空
SS_SCON_3	{content:”xxx”,specificUsers:[10000, 10001]}	code:500	范围为空

SS_SCON_4	{scope:10000,specificUsers:[10000,10001]}	code:500	内容为空
SS_SCON_5	{scope:10000,specificUsers:[10000,10001],content:”xxx”}	code:200	正确用例

测试代码部分实例

```
}
@Test
void testGet() throws Exception {
    // Setup
    // Run the test
    final MockHttpServletResponse response = mockMvc.perform(get("/news/nacos")
                    .accept(MediaType.APPLICATION_JSON))
            .andReturn().getResponse();

    // Verify the results
    assertThat(response.getStatus()).isEqualTo(HttpStatus.OK.value());
    assertThat(response.getContentAsString()).isEqualTo("expectedResponse");
}
```

（2）

8.1.2.2 testTest

（1）用例说明。其中参数均为 MessageForm 的封装属性。

序号	输入参数	期望结果	参数说明
SS_SCON_1		code:500	传空
SS_SCON_2	{scope:10000, content:”xxxx”}	code:500	接收者为空
SS_SCON_3	{content:”xxx”,specificUsers:[10000,10001]}	code:500	范围为空
SS_SCON_4	{scope:10000,specificUsers:[10000,10001]}	code:500	内容为空
SS_SCON_5	{scope:10000,specificUsers:[10000,10001],content:”xxx”}	code:200	正确用例

（2） 测试代码部分实例

```
@Test
void testTest() throws Exception {
    // Setup
    // Run the test
    final MockHttpServletResponse response = mockMvc.perform(post("/news/test")
                    .accept(MediaType.APPLICATION_JSON))
            .andReturn().getResponse();

    // Verify the results
    assertThat(response.getStatus()).isEqualTo(HttpStatus.OK.value());
```

```
    assertThat(response.getContentAsString()).isEqualTo("expectedResponse");
}
```

8.1.2.3 testPostHouse

（1）用例说明。其中参数均为 MessageForm 的封装属性。

序号	输入参数	期望结果	参数说明
SS_SCON_1		code:500	传空
SS_SCON_2	{scope:10000, content:”xxxx”}	code:500	接收者为空
SS_SCON_3	{content:”xxx”,specificUsers:[10000, 10001]}	code:500	范围为空
SS_SCON_4	{scope:10000,specificUsers:[10000,10001]}	code:500	内容为空
SS_SCON_5	{scope:10000,specificUsers:[10000,10001],content:”xxx”}	code:200	正确用例

（2） 测试代码部分实例

```
@Test
void testPostHouse() throws Exception {
    // Setup
    when(mockUserClient.getOneGroup(new HashMap<>())).thenReturn(new Result<>(new HashMap<>()));
    when(mockArticleService.addArticle(new ArticleVo())).thenReturn(new HashMap<>());

    // Run the test
    final MockHttpServletResponse response = mockMvc.perform(post("/news/addArticle")
                    .content("content").contentType(MediaType.APPLICATION_JSON)
                    .accept(MediaType.APPLICATION_JSON))
            .andReturn().getResponse();

    // Verify the results
    assertThat(response.getStatus()).isEqualTo(HttpStatus.OK.value());
    assertThat(response.getContentAsString()).isEqualTo("expectedResponse");
}
```

8.1.2.4 testUpdateHouse

（1）用例说明。其中参数均为 MessageForm 的封装属性。

序号	输入参数	期望结果	参数说明
SS_SCON_1		code:500	传空
SS_SCON_2	{scope:10000, content:”xxxx”}	code:500	接收者为空
SS_SCON_3	{content:”xxx”,specificUsers:[10000,10001]}	code:500	范围为空
SS_SCON_4	{scope:10000,specificUsers:[10000,10001]}	code:500	内容为空
SS_SCON_5	{scope:10000,specificUsers:[10000,10001],content:”xxx”}	code:200	正确用例

（2） 测试代码部分实例

```
@Test
void testUpdateHouse() throws Exception {
    // Setup
    when(mockArticleService.updateArticle(new ArticleVo())).thenReturn(new HashMap<>());

    // Run the test
    final MockHttpServletResponse response = mockMvc.perform(post("/news/updateArticle")
                    .content("content").contentType(MediaType.APPLICATION_JSON)
                    .accept(MediaType.APPLICATION_JSON))
            .andReturn().getResponse();

    // Verify the results
    assertThat(response.getStatus()).isEqualTo(HttpStatus.OK.value());
    assertThat(response.getContentAsString()).isEqualTo("expectedResponse");
}
```

8.1.2.5 testDeleteHouse

（1）用例说明。其中参数均为 MessageForm 的封装属性。

序号	输入参数	期望结果	参数说明
SS_SCON_1		code:500	传空
SS_SCON_2	{scope:10000, content:”xxxx”}	code:500	接收者为空
SS_SCON_3	{content:”xxx”,specificUsers:[10000,10001]}	code:500	范围为空
SS_SCON_4	{scope:10000,specificUsers:[10000,10001]}	code:500	内容为空
SS_SCON_5	{scope:10000,specificUsers:[10000,10001],content:”xxx”}	code:200	正确用例

（2） 测试代码部分实例

```
@Test
```

```
void testDeleteHouse() throws Exception {
    // Setup
    when(mockArticleService.deleteArticle(0)).thenReturn(new HashMap<>());

    // Run the test
    final MockHttpServletResponse response = mockMvc.perform(post("/news/deleteArticle")
                        .content("content").contentType(MediaType.APPLICATION_JSON)
                        .accept(MediaType.APPLICATION_JSON))
                .andReturn().getResponse();

    // Verify the results
    assertThat(response.getStatus()).isEqualTo(HttpStatus.OK.value());
    assertThat(response.getContentAsString()).isEqualTo("expectedResponse");
}
```

8.1.2.6 testSelectHouse

（1）用例说明。其中参数均为 MessageForm 的封装属性。

序号	输入参数	期望结果	参数说明
SS_SCON_1		code:500	传空
SS_SCON_2	{scope:10000, content:”xxxx”}	code:500	接收者为空
SS_SCON_3	{content:”xxx”,specificUsers:[10000, 10001]}	code:500	范围为空
SS_SCON_4	{scope:10000,specificUsers:[10000,10001]}	code:500	内容为空
SS_SCON_5	{scope:10000,specificUsers:[10000,10001],content:”xxx”}	code:200	正确用例

（2） 测试代码部分实例

```
@Test
void testSelectHouse() throws Exception {
    // Setup
    when(mockUserClient.getChildGroupsSql(0L)).thenReturn("result");
    when(mockArticleService.selcetArticle(new HashMap<>())).thenReturn(new HashMap<>());

    // Run the test
    final MockHttpServletResponse response = mockMvc.perform(post("/news/selectArticle")
                        .content("content").contentType(MediaType.APPLICATION_JSON)
                        .accept(MediaType.APPLICATION_JSON))
                .andReturn().getResponse();
```

```
    // Verify the results
    assertThat(response.getStatus()).isEqualTo(HttpStatus.OK.value());
    assertThat(response.getContentAsString()).isEqualTo("expectedResponse");
}
```

8.1.2.7 testSelectHouseForC

（1）用例说明。其中参数均为 MessageForm 的封装属性。

序号	输入参数	期望结果	参数说明
SS_SCON_1		code:500	传空
SS_SCON_2	{scope:10000, content:”xxxx”}	code:500	接收者为空
SS_SCON_3	{content:”xxx”,specificUsers:[10000, 10001]}	code:500	范围为空
SS_SCON_4	{scope:10000,specificUsers:[10000,10001]}	code:500	内容为空
SS_SCON_5	{scope:10000,specificUsers:[10000,10001],content:”xxx”}	code:200	正确用例

（3） 测试代码部分实例

```
}@Test
void testSelectHouseForC() throws Exception {
    // Setup
    when(mockArticleService.selcetArticleForC(new HashMap<>())).thenReturn(new HashMap<>());

    // Run the test
    final MockHttpServletResponse response = mockMvc.perform(post("/news/selectArticleForC")
                    .content("content").contentType(MediaType.APPLICATION_JSON)
                    .accept(MediaType.APPLICATION_JSON))
            .andReturn().getResponse();

    // Verify the results
    assertThat(response.getStatus()).isEqualTo(HttpStatus.OK.value());
    assertThat(response.getContentAsString()).isEqualTo("expectedResponse");
}
```

8.1.2.8 testSelectYourArticle

（1）用例说明。其中参数均为 MessageForm 的封装属性。

序号	输入参数	期望结果	参数说明
SS_SCON_1		code:500	传空
SS_SCON_2	{scope:10000, content:”xxxx”}	code:500	接收者为空
SS_SCON_3	{content:”xxx”,specificUsers:[10000, 10001]}	code:500	范围为空
SS_SCON_4	{scope:10000,specificUsers:[10000,10001]}	code:500	内容为空
SS_SCON_5	{scope:10000,specificUsers:[10000,10001],content:”xxx”}	code:200	正确用例

（2） 测试代码部分实例

```
@Test
void testSelectYourArticle() throws Exception {
    // Setup
    when(mockArticleService.selcetArticle(new HashMap<>())).thenReturn(new HashMap<>());

    // Run the test
    final MockHttpServletResponse response = mockMvc.perform(post("/news/selectYourArticle")
                        .content("content").contentType(MediaType.APPLICATION_JSON)
                        .accept(MediaType.APPLICATION_JSON))
            .andReturn().getResponse();

    // Verify the results
    assertThat(response.getStatus()).isEqualTo(HttpStatus.OK.value());
    assertThat(response.getContentAsString()).isEqualTo("expectedResponse");
}
```

8.1.2.9 testFileUpload

（1）用例说明。其中参数均为 MessageForm 的封装属性。

序号	输入参数	期望结果	参数说明
SS_SCON_1		code:500	传空
SS_SCON_2	{scope:10000, content:”xxxx”}	code:500	接收者为空
SS_SCON_3	{content:”xxx”,specificUsers:[10000, 10001]}	code:500	范围为空
SS_SCON_4	{scope:10000,specificUsers:[10000,10001]}	code:500	内容为空

SS_SCON_5	{scope:10000,specificUsers:[10000,10001],content:”xxx”}	code:200	正确用例

（2） 测试代码部分实例

```
}@Test
void testFileUpload() throws Exception {
    // Setup
    // Run the test
    final MockHttpServletResponse response = mockMvc.perform(multipart("/news/fileupload")
                    .file(new MockMultipartFile("newsPic", "originalFilename",
MediaType.APPLICATION_JSON_VALUE,
                            "content".getBytes()))
                    .accept(MediaType.APPLICATION_JSON))
            .andReturn().getResponse();

    // Verify the results
    assertThat(response.getStatus()).isEqualTo(HttpStatus.OK.value());
    assertThat(response.getContentAsString()).isEqualTo("expectedResponse");
}
```

8.1.2.10 testFileUpload_ThrowsIOException

（1）用例说明。其中参数均为 MessageForm 的封装属性。

序号	输入参数	期望结果	参数说明
SS_SCON_1		code:500	传空
SS_SCON_2	{scope:10000, content:”xxxx”}	code:500	接收者为空
SS_SCON_3	{content:”xxx”,specificUsers:[10000,10001]}	code:500	范围为空
SS_SCON_4	{scope:10000,specificUsers:[10000,10001]}	code:500	内容为空
SS_SCON_5	{scope:10000,specificUsers:[10000,10001],content:”xxx”}	code:200	正确用例

（2） 测试代码部分实例

```
}
```

```
@Test
void testFileUpload_ThrowsIOException() throws Exception {
    // Setup
    // Run the test
    final MockHttpServletResponse response = mockMvc.perform(multipart("/news/fileupload")
```

```
                .file(new MockMultipartFile("newsPic", "originalFilename",
MediaType.APPLICATION_JSON_VALUE,
                        "content".getBytes()))
                .accept(MediaType.APPLICATION_JSON))
        .andReturn().getResponse();

    // Verify the results
    assertThat(response.getStatus()).isEqualTo(HttpStatus.INTERNAL_SERVER_ERROR.value());
    assertThat(response.getContentAsString()).isEqualTo("expectedResponse");
}
```

8.1.2.11 testReport

（1）用例说明。其中参数均为 MessageForm 的封装属性。

序号	输入参数	期望结果	参数说明
SS_SCON_1		code:500	传空
SS_SCON_2	{scope:10000, content:”xxxx”}	code:500	接收者为空
SS_SCON_3	{content:”xxx”,specificUsers:[10000, 10001]}	code:500	范围为空
SS_SCON_4	{scope:10000,specificUsers:[10000,10001]}	code:500	内容为空
SS_SCON_5	{scope:10000,specificUsers:[10000,10001],content:”xxx”}	code:200	正确用例

（2） 测试代码部分实例

```
}
```

```
@Test
void testReport() throws Exception {
    // Setup
    when(mockReportClient.addReport(new HashMap<>())).thenReturn(new Result<>(new HashMap<>()));

    // Run the test
    final MockHttpServletResponse response = mockMvc.perform(post("/news/report")
                .content("content").contentType(MediaType.APPLICATION_JSON)
                .accept(MediaType.APPLICATION_JSON))
        .andReturn().getResponse();

    // Verify the results
    assertThat(response.getStatus()).isEqualTo(HttpStatus.OK.value());
    assertThat(response.getContentAsString()).isEqualTo("expectedResponse");
```

```
    verify(mockReportClient).addReport(new HashMap<>());
}
```

8.1.2.12 testDoAudit

（1）用例说明。其中参数均为MessageForm的封装属性。

序号	输入参数	期望结果	参数说明
SS_SCON_1		code:500	传空
SS_SCON_2	{scope:10000, content:”xxxx”}	code:500	接收者为空
SS_SCON_3	{content:”xxx”,specificUsers:[10000, 10001]}	code:500	范围为空
SS_SCON_4	{scope:10000,specificUsers:[10000,10001]}	code:500	内容为空
SS_SCON_5	{scope:10000,specificUsers:[10000,10001],content:”xxx”}	code:200	正确用例

测试代码部分实例

```
}
```

```
@Test
void testDoAudit() throws Exception {
    // Setup
    // Configure ArticleService.getById(...).
    final Article article = new Article();
    article.setArticleId(0);
    article.setCreateTime(Date.valueOf(LocalDate.of(2020, 1, 1)));
    article.setDeleteMark("deleteMark");
    article.setAuthor(0L);
    article.setTitle("title");
    article.setContent("content");
    article.setStatus(0);
    article.setGroupId(0);
    when(mockArticleService.getById(0)).thenReturn(article);

    when(mockArticleService.updateById(new Article())).thenReturn(false);
    when(mockAuditClient.addAudit(new HashMap<>())).thenReturn(new Result<>(new HashMap<>()));

    // Run the test
    final MockHttpServletResponse response = mockMvc.perform(post("/news/doAudit")
                    .content("content").contentType(MediaType.APPLICATION_JSON)
                    .accept(MediaType.APPLICATION_JSON))
            .andReturn().getResponse();

    // Verify the results
```

```
    assertThat(response.getStatus()).isEqualTo(HttpStatus.OK.value());
    assertThat(response.getContentAsString()).isEqualTo("expectedResponse");
    verify(mockArticleService).updateById(new Article());
    verify(mockAuditClient).addAudit(new HashMap<>());
}
```

8.1.2.13 testGetAuditLog

（1）用例说明。其中参数均为MessageForm的封装属性。

序号	输入参数	期望结果	参数说明
SS_SCON_1		code:500	传空
SS_SCON_2	{scope:10000, content:”xxxx”}	code:500	接收者为空
SS_SCON_3	{content:”xxx”,specificUsers:[10000, 10001]}	code:500	范围为空
SS_SCON_4	{scope:10000,specificUsers:[10000,10001]}	code:500	内容为空
SS_SCON_5	{scope:10000,specificUsers:[10000,10001],content:”xxx”}	code:200	正确用例

测试代码部分实例

}

```
@Test
void testGetAuditLog() throws Exception {
    // Setup
    when(mockAuditClient.getReportList(new HashMap<>())).thenReturn(new Result<>(Arrays.asList("value")));

    // Run the test
    final MockHttpServletResponse response = mockMvc.perform(post("/news/getAuditLog")
                    .content("content").contentType(MediaType.APPLICATION_JSON)
                    .accept(MediaType.APPLICATION_JSON))
            .andReturn().getResponse();

    // Verify the results
    assertThat(response.getStatus()).isEqualTo(HttpStatus.OK.value());
    assertThat(response.getContentAsString()).isEqualTo("expectedResponse");
}
```

8.1.2.14 testGetAuditLog_AuditClientReturnsNoItems

（1）用例说明。其中参数均为MessageForm的封装属性。

序号	输入参数	期望结果	参数说明
SS_SCON_1		code:500	传空
SS_SCON_2	{scope:10000, content:”xxxx”}	code:500	接收者为空
SS_SCON_3	{content:”xxx”,specificUsers:[10000, 10001]}	code:500	范围为空
SS_SCON_4	{scope:10000,specificUsers:[10000,10001]}	code:500	内容为空
SS_SCON_5	{scope:10000,specificUsers:[10000,10001],content:”xxx”}	code:200	正确用例

测试代码部分实例

```
}
```

```
@Test
void testGetAuditLog_AuditClientReturnsNoItems() throws Exception {
    // Setup
    when(mockAuditClient.getReportList(new HashMap<>())).thenReturn(new Result<>(Collections.emptyList()));

    // Run the test
    final MockHttpServletResponse response = mockMvc.perform(post("/news/getAuditLog")
                    .content("content").contentType(MediaType.APPLICATION_JSON)
                    .accept(MediaType.APPLICATION_JSON))
            .andReturn().getResponse();

    // Verify the results
    assertThat(response.getStatus()).isEqualTo(HttpStatus.OK.value());
    assertThat(response.getContentAsString()).isEqualTo("[]");
}
```

8.1.2.15 testGetNewsGroup

（1）用例说明。其中参数均为MessageForm的封装属性。

序号	输入参数	期望结果	参数说明
SS_SCON_1		code:500	传空
SS_SCON_2	{scope:10000, content:”xxxx”}	code:500	接收者为空
SS_SCON_3	{content:”xxx”,specificUsers:[10000, 10001]}	code:500	范围为空
SS_SCON_4	{scope:10000,specificUsers:[10000,10001]}	code:500	内容为空
SS_SCON_5	{scope:10000,specificUsers:[10000,10001],content:”xxx”}	code:200	正确用例

测试代码部分实例

```
}
```

```
@Test
void testGetNewsGroup() throws Exception {
    // Setup
    when(mockUserClient.getChildGroup(0)).thenReturn(new Result<>(new HashMap<>()));

    // Run the test
    final MockHttpServletResponse response = mockMvc.perform(post("/news/getNewsGroup")
                    .accept(MediaType.APPLICATION_JSON))
            .andReturn().getResponse();

    // Verify the results
    assertThat(response.getStatus()).isEqualTo(HttpStatus.OK.value());
    assertThat(response.getContentAsString()).isEqualTo("expectedResponse");
}
```

8.2 Service 层

8.2.1 测试描述

NewsService 模块的 Service 层单元测试主要是通过传入测试人员设置好的参数，查看 Service 层各个函数的具体输出是否符合逻辑，比如返回的数量，返回的属性，是否会抛出异常等等。涉及到的 Service 类包括 ArticleServiceImplTest。

（1）测试对象

Service 层的测试对象即为各个 Service 类中的具体方法，ArticleServiceImpl

（2）测试用例设计

Service 层的职责在于处理具体的业务逻辑。其中 SystemService 模块中的 Service 层主要是为了后台管理服务的，主要业务逻辑为数据库的增删改查，而对于算法没有什么要求，所以对于该 Service 层的单元测试主要是为了校验其返回值的主要属性，包括数量、主键是否正确，至于是否符合业务，需要到接口测试才能体现。

8.2.2 测试用例

对于每个测试对象都给出多个测试用例，以覆盖所有的参数输入情况，包括正常输入、边界值输入、空指输入等传参输入，并给出期望输出，和实例代码。

8.2.2.1 testAddArticle

（1）用例说明。其中参数均为 MessageForm 的封装属性。

序号	输入参数	期望结果	参数说明
SS_SSER_1		flag:false	传空
SS_SSER_2	scope:10000,specificUsers:[10000,10001],content:”xxx”	flag:true,messageId:非空	正确用例
SS_SSER_3	scope:10000,specificUsers:[10000,10001]	flag:false	内容为空

（2） 测试代码部分实例

```
@Test
void testAddArticle() {
    // Setup
    final ArticleVo articleVo = new ArticleVo();
    articleVo.setAuthor(0L);
    articleVo.setTitle("title");
    articleVo.setContent("content");
    articleVo.setArticleId(0);
    articleVo.setCreateTime(Date.valueOf(LocalDate.of(2020, 1, 1)));
    articleVo.setDeleteMark("deleteMark");
    articleVo.setStatus(0);
    articleVo.setGroupId(0);
    articleVo.setPageNo(0);
    articleVo.setPageSize(0);

    // Run the test
    final Map<String, Object> result = articleServiceImplUnderTest.addArticle(articleVo);

    // Verify the results
}
```

8.2.2.1 testUpdateArticle

（1）用例说明。其中参数均为 MessageForm 的封装属性。

序号	输入参数	期望结果	参数说明
SS_SSER_1		flag:false	传空
SS_SSER_2	scope:10000,specificUsers:[10000,10001],content:”xxx”	flag:true,messageId:非空	正确用例
SS_SSER_3	scope:10000,specificUsers:[10000,10001]	flag:false	内容为空

（2） 测试代码部分实例

```
@Test
void testUpdateArticle() {
    // Setup
    final ArticleVo articleVo = new ArticleVo();
    articleVo.setAuthor(0L);
    articleVo.setTitle("title");
    articleVo.setContent("content");
    articleVo.setArticleId(0);
    articleVo.setCreateTime(Date.valueOf(LocalDate.of(2020, 1, 1)));
    articleVo.setDeleteMark("deleteMark");
    articleVo.setStatus(0);
    articleVo.setGroupId(0);
    articleVo.setPageNo(0);
    articleVo.setPageSize(0);

    // Run the test
    final Map<String, Object> result = articleServiceImplUnderTest.updateArticle(articleVo);

    // Verify the results
}
```

8.2.2.1 testDeleteArticle

（1）用例说明。其中参数均为 MessageForm 的封装属性。

序号	输入参数	期望结果	参数说明
SS_SSER_1		flag:false	传空
SS_SSER_2	scope:10000,specificUsers:[10000,10001],content:”xxx”	flag:true,messageId:非空	正确用例
SS_SSER_3	scope:10000,specificUsers:[10000,10001]	flag:false	内容为空

（2） 测试代码部分实例

```
@Test
void testDeleteArticle() {
    // Setup
    // Run the test
    final Map<String, Object> result = articleServiceImplUnderTest.deleteArticle(0);
```

```
    // Verify the results
}
```

8.2.2.1 testSelcetArticle

（1）用例说明。其中参数均为 MessageForm 的封装属性。

序号	输入参数	期望结果	参数说明
SS_SSER_1		flag:false	传空
SS_SSER_2	scope:10000,specificUsers:[10000,10001],content:”xxx”	flag:true,messageId:非空	正确用例
SS_SSER_3	scope:10000,specificUsers:[10000,10001]	flag:false	内容为空

（2） 测试代码部分实例

```
@Test
void testSelcetArticle() {
    // Setup
    final Map<String, Object> selectCondition = new HashMap<>();

    // Run the test
    final Map<String, Object> result = articleServiceImplUnderTest.selcetArticle(selectCondition);

    // Verify the results
}
```

8.2.2.1 testSelcetArticleForC

（1）用例说明。其中参数均为 MessageForm 的封装属性。

序号	输入参数	期望结果	参数说明
SS_SSER_1		flag:false	传空
SS_SSER_2	scope:10000,specificUsers:[10000,10001],content:”xxx”	flag:true,messageId:非空	正确用例
SS_SSER_3	scope:10000,specificUsers:[10000,10001]	flag:false	内容为空

（3） 测试代码部分实例

```
@Test
void testSelcetArticleForC() {
        // Setup
        final Map<String, Object> selectCondition = new HashMap<>();

        // Run the test
        final Map<String, Object> result = articleServiceImplUnderTest.selcetArticleForC(selectCondition);

        // Verify the results
}
```