测试文档

车一晗 181250009

黄婉红 181840096

纳思彧 181250107

王博 181250133

2021 年 6 月 18 日

摘要

本文档为 Heap 小组在 2021 年春学期《软件工程与计算三》课程作业迭代三中为项目所撰写的测试文档。

目录

1 引言

1.1 编写目的

编写本测试方案的目的是为软件开发项目管理者、软件工程师、系统维护工程师、测试工程师提供关于 COIN 知识图谱可视化系统整体系统功能和性能的测试指导。

1.2 对象与范围

本测试方案的合法读者对象为软件开发项目管理者、软件工程师、测试组、系统维护工程师。

1.3 项目背景

在众多知识表示方式中, 知识图谱作为一种语义网络拥有极强的表达能力和建模灵活性：知识图谱是一种语义表示，可以对现实世界中的实体、概念、属性以及它们之间的关系进行建模；其次，知识图谱是其衍生技术的数据交换标准，其本身是一种数据建模的“协议”，相关技术涵盖知识抽取、知识集成、知识管理和知识应用等各个环节。通过构建知识图谱，可以极大化地辅助相关知识的理解。

1.4 名词与术语

COIN：COnstructing and vIsualizing kNowledge graph 知识图谱可视化系统

1.5 测试目标

最终目标是确保软件的功能符合用户的需求, 把尽可能多的问题在发布或交付之前发现并改正。

1. 确保软件完成了它所承诺或公布的功能
2. 确保软件满足了性能的要求
3. 确保软件是健壮的和适应用户环境的
4. 为软件的质量评估提供依据
5. 为软件质量改进和管理提供帮助

1.6 参考文献

1.《软件工程与计算（卷二)》 骆斌 丁二玉 刘钦

2.《软件工程与计算（卷三)》 骆斌 刘嘉 张瑾玉 黄蕾

3.《项目启动文档》, Heap

4.《项目设计文档》, Heap

5.《项目计划文档》, Heap

6.《需求规格说明文档》, Heap

2 测试配套要求

2.1 网络环境

网络硬件：阿里云轻量应用服务器架设 WEB；阿里云服务器 ECS 架设 WEB

网络软件：峰值带宽 5Mbps；峰值带宽 1Mbps

2.2 服务器环境

服务器硬件：1 核 2G，SSD 云盘 40GB；2 核 4G，高效云盘 40GB

服务器软件：CentOS 7.6

2.3 测试手段

基于黑盒测试和白盒测试的单元测试和集成测试，基于 SwaggerUI 的前后端接口交互测试，前端页面测试。

2.4 测试数据

以客户单位具体的业务规则和 COIN 知识图谱可视化系统《需求规格说明文档》，参考 COIN 知识图谱可视化系统《项目启动文档》、《项目设计文档》、《项目计划文档》中规定的运行限制，设计测试用例，作为整个 COIN 知识图谱可视化系统的测试数据。

2.5 测试策略

测试过程按两个步骤进行，即单元测试、集成测试，根据不同阶段测试的侧重点不同，分别介绍测试策略。

一、单元测试

首先按照系统、子系统和模块进行划分，但最终的单元必须是功能模块，或面向对象过程中的若干个类。单元测试是对功能模块进行正确性检验的测试工作，也是后续测试的基础。目的是在于发现各模块内部可能存在的各种差错，因此需要从程序的内部结构出发设计测试用例，着重考虑以下五个方面：

1. 模块接口：对所测模块的数据流进行测试。

2. 局部数据结构：检查不正确或不一致的数据类型说明、使用尚未赋值或尚未初始化的变量、错误的初始值或缺省值。

3. 路径：虽然不可能做到穷举测试，但要设计测试用例查找由于不正确的计算（包括算法错、表达式的符号表示不正确、运算精度不够等）、不正确的比较或不正常的控制流（包括不同数据类型量的相互比较、不适当地修改了循环变量、错误的或不可能的循环终止条件等）而导致的错误。

4. 错误处理：检查模块有没有对预见错误的条件设计比较完善的错误处理功能，保证其逻辑上的正确性。

5. 边界：注意设计数据流、控制流中刚好等于、大于或小于确定的比较值的用例。

二、集成测试

集成测试也叫组装测试或联合测试。通常在单元测试的基础上需要将所有的模块按照设计要求组装成系统，下面是需要考虑的问题：

1. 在把各个模块连接起来的时候，穿越模块接口的数据是否会丢失。
2. 一个模块的功能是否会对另一个模块的功能产生不利的影响。
3. 各个子功能组合起来，能否达到预期要求的父功能。
4. 全局数据结构是否有问题。
5. 单元模块的误差累积起来，是否会放大，从而达到不能接受的程度。
6. 组装时参考采用一次性组装方式

2.6 测试通过准则

在此规定本系统通过测试的准则，即当依据测试用例执行者测试结果与预期结果相符，或测试结果与预期结果虽有不符但不可归咎于应用程序时为测试通过，反之则为测试失败。

3　单元测试用例

3.1 DomainService 系统

3.1.1 创建工作域模块

前提条件：COINApplication 后端正常运行

执行时间：387ms

测试过程：

第一步：模拟后端得到的 controller 传来的数据

第二步：调用 serviceImpl 的 createDomain 方法

第三步：通过 serviceImpl 返回给 controller 的返回值判断是否调用成功

预期输出：

```
assertEquals(expect, actual);
```

3.1.2 删除工作域模块

前提条件：COINApplication 后端正常运行

执行时间：18ms

测试过程：

第一步：模拟后端得到的 controller 传来的数据

第二步：调用 serviceImpl 的 deleteDomain 方法

第三步：通过 serviceImpl 返回给 controller 的返回值判断是否调用成功

预期输出：

```
domainService.deleteDomain(expect);
```

3.1.3 更新工作域模块

前提条件：COINApplication 后端正常运行

执行时间：18ms

测试过程：

第一步：模拟后端得到的 controller 传来的数据

第二步：调用 serviceImpl 的 updateDomain 方法

第三步：通过 serviceImpl 返回给 controller 的返回值判断是否调用成功

预期输出：

```
assertEquals(actual.getId( ), given.getId( ));
assertEquals(actual.getName( ), given.getName( ));
```

3.1.4 获取指定工作域模块

前提条件：COINApplication 后端正常运行

执行时间：387ms

测试过程：

第一步：模拟后端得到的 controller 传来的数据

第二步：调用 serviceImpl 的 getDomainById 方法

第三步：通过 serviceImpl 返回给 controller 的返回值判断是否调用成功

预期输出：

```
assertEquals(2,actual.getId( ));
assertEquals("coin", actual.getName( ));
```

3.1.5 获取所有工作域模块

前提条件：COINApplication 后端正常运行

执行时间：14ms

测试过程：

第一步：模拟后端得到的 controller 传来的数据

第二步：调用 serviceImpl 的 getAllDomain 方法

第三步：通过 serviceImpl 返回给 controller 的返回值判断是否调用成功

预期输出：

```
assertEquals(2,domains.size( ));
```

3.1.6 获取模板工作域模块

前提条件：COINApplication 后端正常运行

执行时间：17ms

测试过程：

第一步：模拟后端得到的 controller 传来的数据

第二步：调用 serviceImpl 的 getTemplate 方法

第三步：通过 serviceImpl 返回给 controller 的返回值判断是否调用成功

预期输出：

```
assertEquals("宝可梦", domain.getName());
```

3.2 EntityService 系统

3.2.1 创建节点模块

前提条件：COINApplication 后端正常运行

执行时间：13ms

测试过程：

第一步：模拟后端得到的 controller 传来的数据

第二步：调用 serviceImpl 的 createNode 方法

第三步：通过 serviceImpl 返回给 controller 的返回值判断是否调用成功

预期输出：

```
assertEquals(expect, actual);
```

3.2.2 获取指定节点模块

前提条件：COINApplication 后端正常运行

执行时间：13ms

测试过程：

第一步：模拟后端得到的 controller 传来的数据

第二步：调用 serviceImpl 的 getNodeByDomainId 方法

第三步：通过 serviceImpl 返回给 controller 的返回值判断是否调用成功

预期输出：

```
assertEquals(1, entities.size( ));
```

3.2.3 更新节点模块

前提条件：COINApplication 后端正常运行

执行时间：387ms

测试过程：

第一步：模拟后端得到的 controller 传来的数据

第二步：调用 serviceImpl 的 updateNode 方法

第三步：通过 serviceImpl 返回给 controller 的返回值判断是否调用成功

预期输出：

```
assertEquals(expect, actual);
```

3.2.4 删除节点模块

前提条件：COINApplication 后端正常运行

执行时间：13ms

测试过程：

第一步：模拟后端得到的 controller 传来的数据

第二步：调用 serviceImpl 的 deleteNode 方法

第三步：通过 serviceImpl 返回给 controller 的返回值判断是否调用成功

预期输出：

```
assertTrue(judge);
```

3.2.5 按名查找节点模块

前提条件：COINApplication 后端正常运行

执行时间：17ms

测试过程：

第一步：模拟后端得到的 controller 传来的数据

第二步：调用 serviceImpl 的 findByName 方法

第三步：通过 serviceImpl 返回给 controller 的返回值判断是否调用成功

预期输出：

```
assertEquals("wb", result1.get(0).getName());
assertEquals("w", result2.get(0).getName());
```

3.2.6 计数节点模块

前提条件：COINApplication 后端正常运行

执行时间：15ms

测试过程：

第一步：模拟后端得到的 controller 传来的数据

第二步：调用 serviceImpl 的 countAllEntity 方法

第三步：通过 serviceImpl 返回给 controller 的返回值判断是否调用成功

预期输出：

```
assertEquals(1, result);
```

3.2.7 按类型计数节点模块

前提条件：COINApplication 后端正常运行

执行时间：20ms

测试过程：

第一步：模拟后端得到的 controller 传来的数据

第二步：调用 serviceImpl 的 countEntitiesByType 方法

第三步：通过 serviceImpl 返回给 controller 的返回值判断是否调用成功

预期输出：

```
assertEquals(1, entityService.countEntitiesByType(0, "test"));
```

3.2.8 更新节点力导坐标模块

前提条件：COINApplication 后端正常运行

执行时间：13ms

测试过程：

第一步：模拟后端得到的 controller 传来的数据

第二步：调用 serviceImpl 的 updateXY 方法

第三步：通过 serviceImpl 返回给 controller 的返回值判断是否调用成功

预期输出：

```
assertTrue(judge);
```

3.2.9 更新节点排版坐标模块

前提条件：COINApplication 后端正常运行

执行时间：16ms

测试过程：

第一步：模拟后端得到的 controller 传来的数据

第二步：调用 serviceImpl 的 updateComposeXY 方法

第三步：通过 serviceImpl 返回给 controller 的返回值判断是否调用成功

预期输出：

```
assertTrue(judge);
```

3.2.10 按类型取节点模块

前提条件：COINApplication 后端正常运行

执行时间：13ms

测试过程：

第一步：模拟后端得到的 controller 传来的数据

第二步：调用 serviceImpl 的 getNodeByType 方法

第三步：通过 serviceImpl 返回给 controller 的返回值判断是否调用成功

预期输出：

```
assertEquals(1, result.size());
assertEquals("test",result.get(0).getType());
assertEquals(2,result.get(0).getDomainId());
```

3.2.11 更新节点类型模块

前提条件：COINApplication 后端正常运行

执行时间：23ms

测试过程：

第一步：模拟后端得到的 controller 传来的数据

第二步：调用 serviceImpl 的 updateType 方法

第三步：通过 serviceImpl 返回给 controller 的返回值判断是否调用成功

预期输出：

```
assertTrue(judge);
```

3.3 FileService 系统

3.3.1 导入 CSV 文件创建图谱模块

前提条件：COINApplication 后端正常运行

执行时间：327ms

测试过程：

第一步：模拟后端得到的 controller 传来的数据

第二步：调用 serviceImpl 的 createGraphByCsv 方法

第三步：通过 serviceImpl 返回给 controller 的返回值判断是否调用成功

预期输出：

```
assertTrue(judge);
```

3.3.2 图谱导出 XML 文件模块

前提条件：COINApplication 后端正常运行

执行时间：15ms

测试过程：

第一步：模拟后端得到的 controller 传来的数据

第二步：调用 serviceImpl 的 exportGraphXml 方法

第三步：通过 serviceImpl 返回给 controller 的返回值判断是否调用成功

预期输出：

```
assertTrue(judge);
```

3.3.3 图谱删除文件模块

前提条件：COINApplication 后端正常运行

执行时间：16ms

测试过程：

第一步：模拟后端得到的 controller 传来的数据

第二步：调用 serviceImpl 的 deleteFile 方法

第三步：通过 serviceImpl 返回给 controller 的返回值判断是否调用成功

预期输出：

```
assertTrue(judge);
```

3.4 RelationshipService 系统

3.4.1 创建连接模块

前提条件：COINApplication 后端正常运行

执行时间：12ms

测试过程：

第一步：模拟后端得到的 controller 传来的数据

第二步：调用 serviceImpl 的 createLink 方法

第三步：通过 serviceImpl 返回给 controller 的返回值判断是否调用成功

预期输出：

```
assertEquals((Long) 0L, (Long) actual.getFromId( ));
assertEquals((Long) 0L, (Long) actual.getToId( ));
assertEquals("link", actual.getName( ));
```

3.4.2 更新连接模块

前提条件：COINApplication 后端正常运行

执行时间：13ms

测试过程：

第一步：模拟后端得到的 controller 传来的数据

第二步：调用 serviceImpl 的 updateLink 方法

第三步：通过 serviceImpl 返回给 controller 的返回值判断是否调用成功

预期输出：

```
assertEquals(test.getFromId( ), actual.getFromId( ));
assertEquals(test.getToId( ), actual.getToId( ));
assertEquals(test.getName( ), actual.getName( ));
```

3.4.3 删除连接模块

前提条件：COINApplication 后端正常运行

执行时间：13ms

测试过程：

第一步：模拟后端得到的 controller 传来的数据

第二步：调用 serviceImpl 的 deleteLink 方法

第三步：通过 serviceImpl 返回给 controller 的返回值判断是否调用成功

预期输出：

```
assertTrue(judge);
```

3.4.4 获取指定连接模块

前提条件：COINApplication 后端正常运行

执行时间：342ms

测试过程：

第一步：模拟后端得到的 controller 传来的数据

第二步：调用 serviceImpl 的 getLinkByDomainId 方法

第三步：通过 serviceImpl 返回给 controller 的返回值判断是否调用成功

预期输出：

```
assertEquals(1, relationships.size( ));
```

3.4.5 计数连接模块

前提条件：COINApplication 后端正常运行

执行时间：25ms

测试过程：

第一步：模拟后端得到的 controller 传来的数据

第二步：调用 serviceImpl 的 countAllLink 方法

第三步：通过 serviceImpl 返回给 controller 的返回值判断是否调用成功

预期输出：

```
assertEquals(result,1);
```

3.4.6 按名获取连接模块

前提条件：COINApplication 后端正常运行

执行时间：27ms

测试过程：

第一步：模拟后端得到的 controller 传来的数据

第二步：调用 serviceImpl 的 getLinkByName 方法

第三步：通过 serviceImpl 返回给 controller 的返回值判断是否调用成功

预期输出：

```
assertEquals(1,result.size());
assertEquals("test",result.get(0).getName());
assertEquals(0,result.get(0).getDomainId());
```

3.4.7 按节点获取连接模块

前提条件：COINApplication 后端正常运行

执行时间：18ms

测试过程：

第一步：模拟后端得到的 controller 传来的数据

第二步：调用 serviceImpl 的 getGraphScreen 方法

第三步：通过 serviceImpl 返回给 controller 的返回值判断是否调用成功

预期输出：

```
assertEquals(1,result.size());
assertEquals("test",result.get(0).getName());
assertEquals(0,result.get(0).getDomainId());
```

3.5 TypeService 系统

3.5.1 插入类型模块

前提条件：COINApplication 后端正常运行

执行时间：362ms

测试过程：

第一步：模拟后端得到的 controller 传来的数据

第二步：调用 serviceImpl 的 insertType 方法

第三步：通过 serviceImpl 返回给 controller 的返回值判断是否调用成功

预期输出：

```
assertEquals("#000000",result);
```

3.5.2 删除类型模块

前提条件：COINApplication 后端正常运行

执行时间：16ms

测试过程：

第一步：模拟后端得到的 controller 传来的数据

第二步：调用 serviceImpl 的 deleteType 方法

第三步：通过 serviceImpl 返回给 controller 的返回值判断是否调用成功

预期输出：

```
assertNull(typeService.searchColorByType(0,"test"));
```

3.5.3 按类型获取颜色模块

前提条件：COINApplication 后端正常运行

执行时间：23ms

测试过程：

第一步：模拟后端得到的 controller 传来的数据

第二步：调用 serviceImpl 的 searchColorByType 方法

第三步：通过 serviceImpl 返回给 controller 的返回值判断是否调用成功

预期输出：

```
assertNull(typeService.searchColorByType(0,"test"));
```

3.5.4 获取所有类型模块

前提条件：COINApplication 后端正常运行

执行时间：27ms

测试过程：

第一步：模拟后端得到的 controller 传来的数据

第二步：调用 serviceImpl 的 searchAll 方法

第三步：通过 serviceImpl 返回给 controller 的返回值判断是否调用成功

预期输出：

```
assertEquals("test",result.get(0));
```

3.6 QuestionService 系统

3.6.1 删除字典模块

前提条件：COINApplication 后端正常运行

执行时间：367ms

测试过程：

第一步：模拟后端得到的 controller 传来的数据

第二步：调用 serviceImpl 的 clean 方法

第三步：通过 serviceImpl 返回给 controller 的返回值判断是否调用成功

预期输出：

```
assertEquals(byCheck,nowCheck+entities.size());
```

3.6.2 插入字典模块

前提条件：COINApplication 后端正常运行

执行时间：27ms

测试过程：

第一步：模拟后端得到的 controller 传来的数据

第二步：调用 serviceImpl 的 addDict 方法

第三步：通过 serviceImpl 返回给 controller 的返回值判断是否调用成功

预期输出：

```
assertEquals("皮卡丘  15 nr",result);
```

3.6.3 执行脚本模块

前提条件：COINApplication 后端正常运行

执行时间：12ms

测试过程：

第一步：模拟后端得到的 controller 传来的数据

第二步：调用 serviceImpl 的 dealByPython 方法

第三步：通过 serviceImpl 返回给 controller 的返回值判断是否调用成功

预期输出：

```
assertNotSame(results,null);
```

3.6.4 问题解答模块

前提条件：COINApplication 后端正常运行

执行时间：12ms

测试过程：

第一步：模拟后端得到的 controller 传来的数据

第二步：调用 serviceImpl 的 getAnswer 方法

第三步：通过 serviceImpl 返回给 controller 的返回值判断是否调用成功

预期输出：

```
assertEquals(answers,result);
```

3.7 UserService 系统

3.7.1 登陆模块

前提条件：COINApplication 后端正常运行

执行时间：10ms

测试过程：

第一步：模拟后端得到的 controller 传来的数据

第二步：调用 serviceImpl 的 login 方法

第三步：通过 serviceImpl 返回给 controller 的返回值判断是否调用成功

预期输出：

```
assertNotNull(a,null);
```

3.7.2 注册模块

前提条件：COINApplication 后端正常运行

执行时间：10ms

测试过程：

第一步：模拟后端得到的 controller 传来的数据

第二步：调用 serviceImpl 的 register 方法

第三步：通过 serviceImpl 返回给 controller 的返回值判断是否调用成功

预期输出：

```
assertNotNull(a,null);
```

3.7.3 获取用户 info 模块

前提条件：COINApplication 后端正常运行

执行时间：11ms

测试过程：

第一步：模拟后端得到的 controller 传来的数据

第二步：调用 serviceImpl 的 getUserInfo 方法

第三步：通过 serviceImpl 返回给 controller 的返回值判断是否调用成功

预期输出：

```
assertEquals(u,userInfoVO);
```

3.7.4 更新用户 info 模块

前提条件：COINApplication 后端正常运行

执行时间：11ms

测试过程：

第一步：模拟后端得到的 controller 传来的数据

第二步：调用 serviceImpl 的 updateInfo 方法

第三步：通过 serviceImpl 返回给 controller 的返回值判断是否调用成功

预期输出：

```
assert (userInfoVO.getIsVip());
```

3.7.5 更新用户 Avatar 模块

前提条件：COINApplication 后端正常运行

执行时间：11ms

测试过程：

第一步：模拟后端得到的 controller 传来的数据

第二步：调用 serviceImpl 的 updateAvatar 方法

第三步：通过 serviceImpl 返回给 controller 的返回值判断是否调用成功

预期输出：

```
assertEquals("88888888",userInfoVO.getAvatar());
```

3.7.6 更新密码模块

前提条件：COINApplication 后端正常运行

执行时间：11ms

测试过程：

第一步：模拟后端得到的 controller 传来的数据

第二步：调用 serviceImpl 的 updatePassword 方法

第三步：通过 serviceImpl 返回给 controller 的返回值判断是否调用成功

预期输出：

```
assertNotNull(a,null);
```

3.7.7 重置密码模块

前提条件：COINApplication 后端正常运行

执行时间：11ms

测试过程：

第一步：模拟后端得到的 controller 传来的数据

第二步：调用 serviceImpl 的 resetPassword 方法

第三步：通过 serviceImpl 返回给 controller 的返回值判断是否调用成功

预期输出：

```
assertNotNull(a,null);
```

3.7.8 注销用户模块

前提条件：COINApplication 后端正常运行

执行时间：11ms

测试过程：

第一步：模拟后端得到的 controller 传来的数据

第二步：调用 serviceImpl 的 disableUser 方法

第三步：通过 serviceImpl 返回给 controller 的返回值判断是否调用成功

预期输出：

```
assertNull(a);
```

3.7.9 管理员更新用户模块

前提条件：COINApplication 后端正常运行

执行时间：11ms

测试过程：

第一步：模拟后端得到的 controller 传来的数据

第二步：调用 serviceImpl 的 updateUserInfoAdmin 方法

第三步：通过 serviceImpl 返回给 controller 的返回值判断是否调用成功

预期输出：

```
assert (A.getIsVip());
```

3.7.10 管理员新增用户模块

前提条件：COINApplication 后端正常运行

执行时间：11ms

测试过程：

第一步：模拟后端得到的 controller 传来的数据

第二步：调用 serviceImpl 的 addUser 方法

第三步：通过 serviceImpl 返回给 controller 的返回值判断是否调用成功

预期输出：

```
assertNotNull(a,null);
```

3.7.11 管理员删除用户模块

前提条件：COINApplication 后端正常运行

执行时间：11ms

测试过程：

第一步：模拟后端得到的 controller 传来的数据

第二步：调用 serviceImpl 的 deleteUser 方法

第三步：通过 serviceImpl 返回给 controller 的返回值判断是否调用成功

预期输出：

```
assertNull(a);
```

3.7.12 用户会员模块

前提条件：COINApplication 后端正常运行

执行时间：11ms

测试过程：

第一步：模拟后端得到的 controller 传来的数据

第二步：调用 serviceImpl 的 setVipUser 方法

第三步：通过 serviceImpl 返回给 controller 的返回值判断是否调用成功

预期输出：

```
assert (userInfoVO.getIsVip());
```

4 集成测试用例

迭代三的集成测试用例大多结构类似并高度集中在对 Controller 类的测试，主要分为两种测试策略：

1. 使用@SpringBootTest 进行测试，使用 TestRestTemplate 测试 Controller 类：

・在测试类上加入@RunWith(SpringRunner.class)与@SpringBootTest 注解

・编写测试方法并添加@Test 注解

示例代码如下：

```
@RunWith(SpringRunner.class)
@SpringBootTest(webEnvironment=SpringBootTest.WebEnvironment.DEFINED_PORT)
public class LoginControllerTest {
	@Autowired
	private TestRestTemplate testRestTemplate;
	@Test
	public void login() {
//		String result =
testRestTemplate.postForObject(/coinservice/user/login,String.class);
//		Assert.assertTrue(result.contains("token"));
	}

	@Test
	public void info() {
//		String result = testRestTemplate.postForObject(/coinservice/user/info,String.class);
//		Assert.assertTrue(result.contains("token"));

	}
}
```

2. 使用@WebMvcTest 注解测试

・在测试类上加入@RunWith(SpringRunner.class)与@WebMvcTest 注解

・使用 MockMVC 对象测试

示例代码如下：

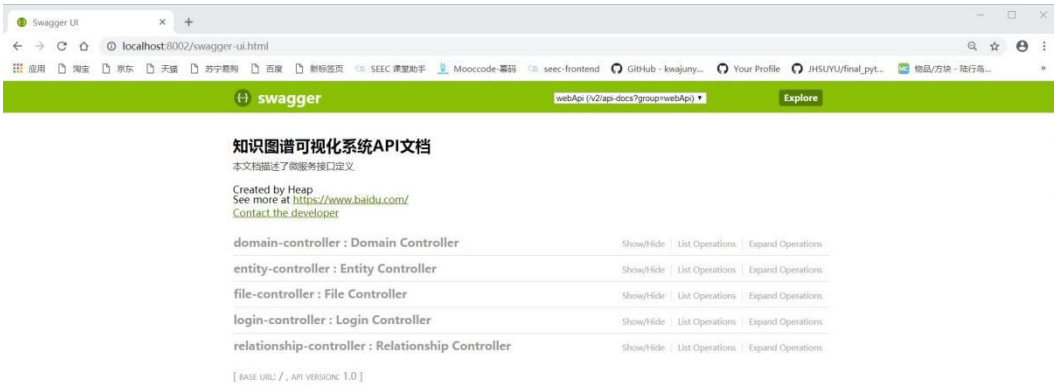
```
package com.heap.coinservice.controller;

import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.autoconfigure.web.servlet.WebMvcTest;
import org.springframework.test.context.junit4.SpringRunner;
import org.springframework.test.web.servlet.MockMvc;

import static org.junit.Assert.*;
@RunWith(SpringRunner.class)
@WebMvcTest(AdminController.class)
public class LoginControllerTest {
    @Autowired
    private MockMvc mockMvc;
    @Test
    public void login() {
        //
mockMvc.perform(MockMvcRequestBuilders.post("/coinservice/user/login")).andExpect(Moc
kMvcResultMatchers.status().isOk());
        //
mockMvc.perform(MockMvcRequestBuilders.post("/coinservice/user/login")).andExpect(Moc
kMvcResultMatchers.content().string(Matchers.containsString("token")));
    }

    @Test
    public void info() {
        //
mockMvc.perform(MockMvcRequestBuilders.post("/coinservice/user/info")).andExpect(Mock
MvcResultMatchers.status().isOk());
        //
mockMvc.perform(MockMvcRequestBuilders.post("/coinservice/user/info")).andExpect(Mock
MvcResultMatchers.content().string(Matchers.containsString("token")));
    }
}
```

通过集成测试验证了代码库的整个调用路径，测试了完整的 SpringMVC 流程，即从 URL 请求到控制器处理，再到视图渲染。具体测试代码在此不做赘述。

5　前后端接口交互测试

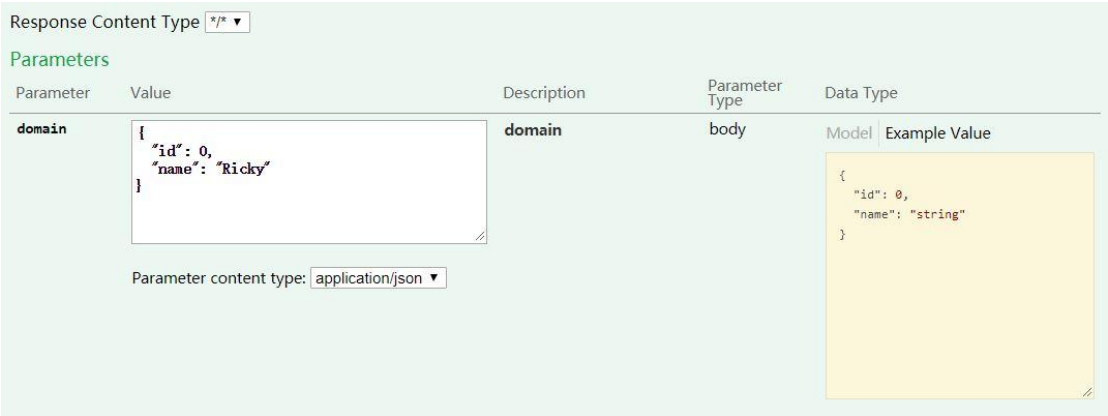
5.1 swagger 说明

通过配置 swagger，可以用于模拟前端向后端的传输接口



5.2 DomainController 测试

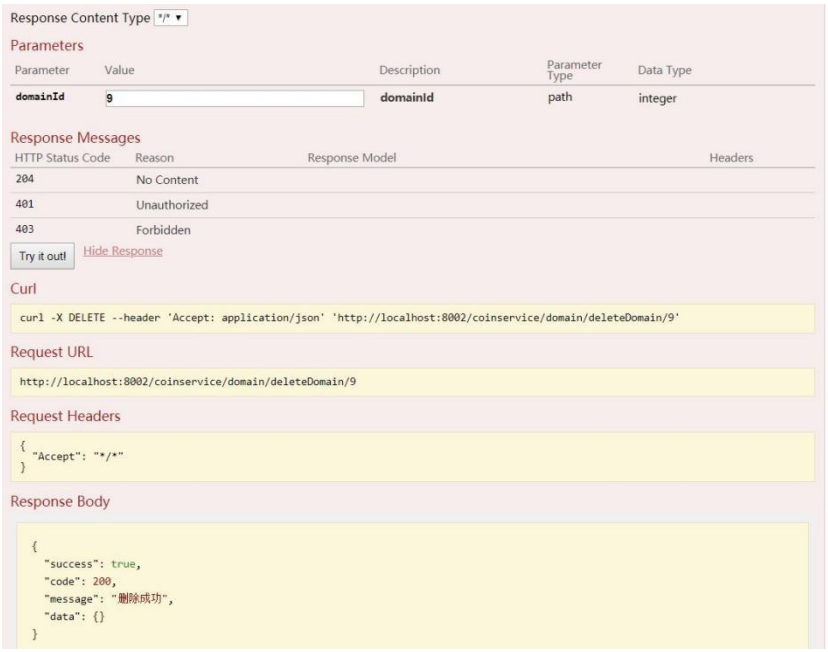
5.2.1 新增域



Response Body


```
{
  "success": true,
  "code": 200,
  "message": "创建成功",
  "data": {}
}
```


5.1.2 删除域



5.2.3 修改域

Response Body


```
{
  "success": true,
  "code": 200,
  "message": "成功",
  "data": {
    "domain": {
      "id": 10,
      "name": "Astley"
    }
  }
}
```


5.2.4 获取域

Response Body


```
{
  "success": true,
  "code": 200,
  "message": "成功",
  "data": {
    "domain": [
      {
        "id": 1,
        "name": "azg"
      },
      {
        "id": 8,
        "name": "as"
      },
      {
        "id": 10,
        "name": "Astley"
      }
    ]
  }
```

5.3 节点测试

5.3.1 新增节点

Curl

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' -d '{ \
   "bgColor": "#33ccff", \
   "domainId": 10, \
   "id": 0, \
   "name": "汤姆", \
   "shape": 0 \
 }' 'http://localhost:8002/coinservice/entity/createNode'
```

Request URL

```
http://localhost:8002/coinservice/entity/createNode
```

Request Headers

```
{
 "Accept": "*/*"
}
```

Response Body


```
{
  "success": true,
  "code": 200,
  "message": "成功",
  "data": {
    "entity": {
      "id": 1,
      "name": "汤姆",
      "bgColor": "#33ccff",
      "shape": 0,
      "domainId": 10
    }
  }
}
```


5.3.2 删除节点

Curl

```
curl -X DELETE --header 'Content-Type: application/json' --header 'Accept: application/json' -d '{ \
   "bgColor": "#333333", \
   "domainId": 8, \
   "id": 0, \
   "name": "哈哈哈", \
   "shape": 1 \
 }' 'http://localhost:8002/coinservice/entity/deleteNode'
```

Request URL

```
http://localhost:8002/coinservice/entity/deleteNode
```

Request Headers

```
{
 "Accept": "*/*"
}
```

Response Body


```
{
  "success": true,
  "code": 200,
  "message": "成功",
  "data": {}
}
```

5.3.3 修改节点

Curl

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' -d '{ \
       "id": 4, \
       "name": "斯派克", \
       "bgColor": "#33ccff", \
       "shape": 0, \
       "domainId": 10 \
     }' 'http://localhost:8002/coinservice/entity/updateNode'
```

Request URL

```
http://localhost:8002/coinservice/entity/updateNode
```

Request Headers

```
{
  "Accept": "*/*"
}
```

Response Body


```
{
  "success": true,
  "code": 200,
  "message": "成功",
  "data": {
    "newEntity": {
      "id": 4,
      "name": "斯派克",
      "bgColor": "#33ccff",
      "shape": 0,
      "domainId": 10
    }
  }
}
```


5.3.4 显示节点

Curl

```
curl -X GET --header 'Accept: application/json' 'http://localhost:8002/coinservice/entity/getNodesByDomainId/10'
```

Request URL

```
http://localhost:8002/coinservice/entity/getNodesByDomainId/10
```

Request Headers

```
{
  "Accept": "*/*"
}
```

Response Body


```
"nodesList": [
  {
    "id": 1,
    "name": "汤姆",
    "bgColor": "#33ccff",
    "shape": 0,
    "domainId": 10
  },
  {
    "id": 2,
    "name": "杰瑞",
    "bgColor": "#33ccff",
    "shape": 0,
    "domainId": 10
  },
  {
    "id": 3,
    "name": "泰菲",
    "bgColor": "#33ccff",
    "shape": 0,
```

5.3.5 统计节点个数

Parameters

Parameter	Value	Description	Parameter Type	Data Type
domainId	15	domainId	path	integer

Response Messages

HTTP Status Code	Reason	Response Model	Headers
401	Unauthorized		
403	Forbidden		
404	Not Found		

Try it out! Hide Response

Curl

```
curl -X GET --header 'Accept: application/json' 'http://localhost:8002/coinservice/entity/countNode/15'
```

Request URL

```
http://localhost:8002/coinservice/entity/countNode/15
```

Request Headers

```
{
  "Accept": "*/*"
}
```

Response Body


```
{
  "success": true,
  "code": 200,
  "message": "成功",
  "data": {
    "total": 2
  }
}
```

5.3.6 显示节点类型

Parameter	Value	Description	Parameter Type	Data Type
domainId	15	domainId	query	integer

Response Messages

HTTP Status Code	Reason	Response Model	Headers
401	Unauthorized		
403	Forbidden		
404	Not Found		

Try it out! Hide Response

Curl

```
curl -X GET --header 'Accept: application/json' 'http://localhost:8002/coinservice/entity/getTypes/{domainId}?domainId=15'
```

Request URL

```
http://localhost:8002/coinservice/entity/getTypes/{domainId}?domainId=15
```

Request Headers

```
{
  "Accept": "*/*"
}
```

Response Body


```
{
  "success": true,
  "code": 200,
  "message": "成功",
  "data": {
    "Types": [
      "a",
      "b"
    ]
  }
}
```

5.3.7 更新节点位置

Curl

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' -d '[ \
   { \
     "bgColor": "string", \
     "composeX": 0, \
     "composeY": 0, \
     "description": "string", \
     "domainId": 15, \
     "fontSize": 40, \
     "id": 88, \
     "name": "a", \
     "r": 40, \
     "shape": 0, \
     "type": "char", \
     "x": 4.56, \
     "y": 4.56 \
   } \
 ]' 'http://localhost:8002/coinservice/entity/updateXY'
```

Request URL

```
http://localhost:8002/coinservice/entity/updateXY
```

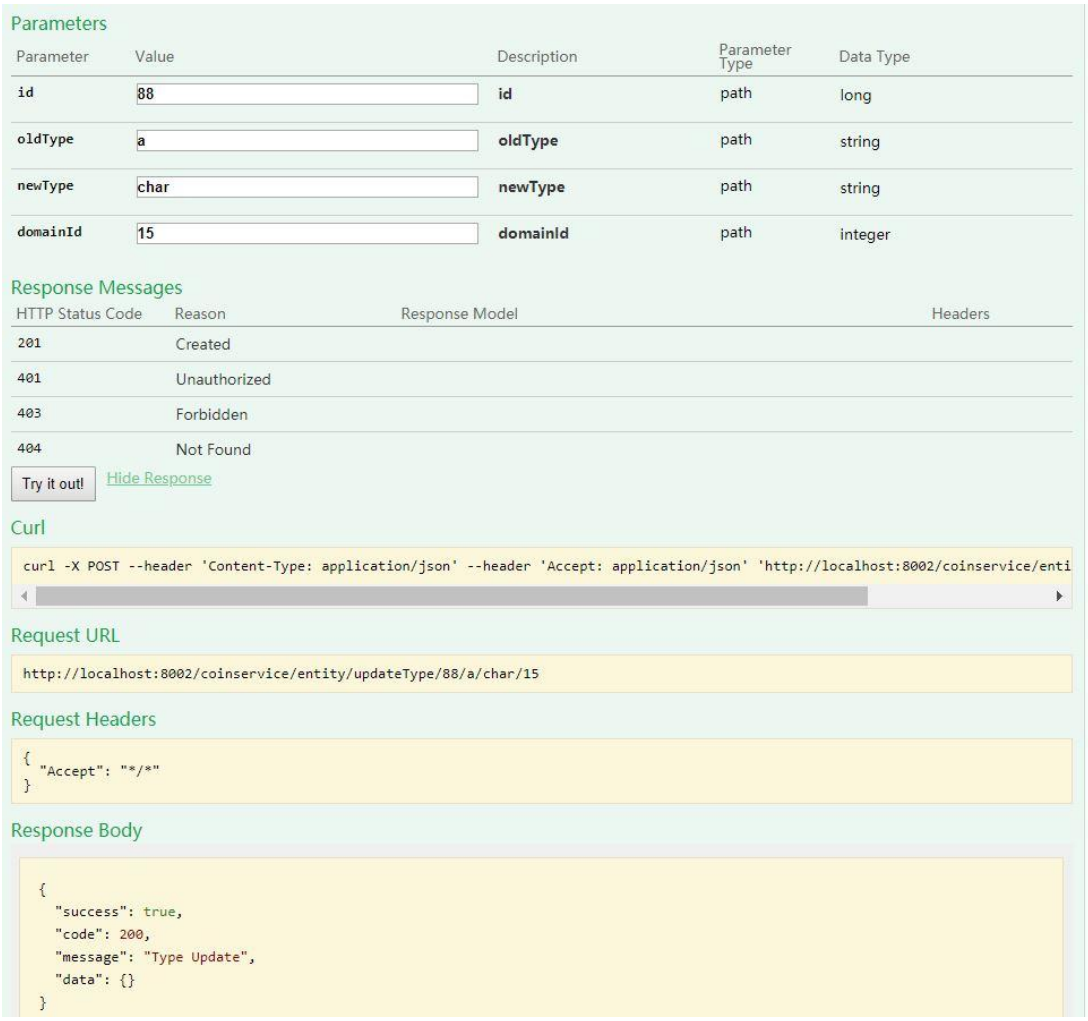
Request Headers

```
{
  "Accept": "*/*"
}
```

Response Body


```
{
  "success": true,
  "code": 200,
  "message": "location save!",
  "data": {}
}
```

5.3.8 更新节点类型



Parameters

Parameter	Value	Description	Parameter Type	Data Type
id	88	id	path	long
oldType	a	oldType	path	string
newType	char	newType	path	string
domainId	15	domainId	path	integer

Response Messages

HTTP Status Code	Reason	Response Model	Headers
201	Created		
401	Unauthorized		
403	Forbidden		
404	Not Found		

Try it out! Hide Response

Curl

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' 'http://localhost:8002/coinservice/enti
```

Request URL

```
http://localhost:8002/coinservice/entity/updateType/88/a/char/15
```

Request Headers

```
{
  "Accept": "*/*"
}
```

Response Body

```
{
  "success": true,
  "code": 200,
  "message": "Type Update",
  "data": {}
}
```

5.3.9 搜索节点

Try it out! Hide Response

Curl

```
curl -X GET --header 'Accept: application/json' 'http://localhost:8002/coinservice/relationship/searchLink/8/%E5%8A%A0%E6%88%90'
```

Request URL

```
http://localhost:8002/coinservice/relationship/searchLink/8/%E5%8A%A0%E6%88%90
```

Request Headers

```
{
  "Accept": "*/*"
}
```

Response Body


```
searchLinkName : [
 {
   "id": 2,
   "name": "加成",
   "type": 1,
   "fromId": 120,
   "toId": 60,
   "domainId": 8,
   "startEntity": {
     "id": 120,
     "name": "烯烃",
     "bgColor": "#C70039",
     "shape": 0,
     "domainId": 8,
     "composeX": 0,
     "composeY": 0,
     "r": 40,
     "type": "烃",
     "description": "",
     "fontSize": 20,
     "y": 282.5673716650336,
```


5.4 关系测试

5.4.1 新增关系

Response Content Type */* ▾

Parameters

Parameter	Value	Description	Parameter Type	Data Type
fromId	1	**fromId**	path	long
toId	2	**toId**	path	long
name	抓	**name**	path	string

Response Body


```
"relationship": {
  "id": 0,
  "name": "抓",
  "type": 1,
  "fromId": 1,
  "toId": 2,
  "domainId": 10,
  "startEntity": {
    "id": 1,
    "name": "汤姆",
    "bgColor": "#33ccff",
    "shape": 0,
    "domainId": 10
  },
  "endEntity": {
    "id": 2,
    "name": "杰瑞",
    "bgColor": "#33ccff",
    "shape": 0,
    "domainId": 10
```


5.4.2 删除关系

Curl

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' -d '{ \
       "id": 20, \
       "name": "Azure", \
       "type": 1, \
       "fromId": 1, \
       "toId": 3, \
       "domainId": 10, \
       "startEntity": { \
         "id": 1, \
         "name": "汤姆", \
         "bgColor": "#33ccff", \
         "shape": 0, \
         "domainId": 10 \
       }, \
       "endEntity": { \
         "id": 3, \
         "name": "泰菲", \
         "bgColor": "#33ccff", \
         "shape": 0, \
         "domainId": 10 \
       } \
     }' 'http://localhost:8002/coinservice/relationship/deleteLink'
```

Request URL

```
http://localhost:8002/coinservice/relationship/deleteLink
```

Request Headers

```
{
  "Accept": "*/*"
}
```

Response Body


```
{
  "success": true,
  "code": 200,
  "message": "成功",
  "data": {}
}
```

5.4.3 修改关系

Curl

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' -d '{ \
       "id": 1, \
       "name": "叔侄", \
       "type": 1, \
       "fromId": 2, \
       "toId": 3, \
       "domainId": 10, \
       "startEntity": { \
         "id": 2, \
         "name": "杰瑞", \
         "bgColor": "#33ccff", \
         "shape": 0, \
         "domainId": 10 \
       }, \
       "endEntity": { \
         "id": 3, \
         "name": "泰菲", \
         "bgColor": "#33ccff", \
         "shape": 0, \
         "domainId": 10 \
       } \
     }' 'http://localhost:8002/coinservice/relationship/updateLink'
```

Request URL

```
http://localhost:8002/coinservice/relationship/updateLink
```

Request Headers

```
{
  "Accept": "*/*"
}
```

Response Body


```
{
  "success": true,
  "code": 200,
  "message": "成功",
  "data": {
    "newRelationship": null
  }
}
```

5.4.4 显示关系

Request URL

```
http://localhost:8002/coinservice/relationship/getLinkByDomainId/10
```

Request Headers

```
{
  "Accept": "*/*"
}
```

Response Body


```
        "name": "抓",
        "type": 1,
        "fromId": 1,
        "toId": 2,
        "domainId": 10,
        "startEntity": {
          "id": 1,
          "name": "汤姆",
          "bgColor": "#33ccff",
          "shape": 0,
          "domainId": 10
        },
        "endEntity": {
          "id": 2,
          "name": "杰瑞",
          "bgColor": "#33ccff",
          "shape": 0,
          "domainId": 10
        }
      },
```


Response Code

```
200
```

5.4.5 统计关系

Parameter	Value	Description	Parameter Type	Data Type
domainId	15	domainId	path	integer

Response Messages

HTTP Status Code	Reason	Response Model	Headers
401	Unauthorized		
403	Forbidden		
404	Not Found		

Try it out! Hide Response

Curl

```
curl -X GET --header 'Accept: application/json' 'http://localhost:8002/coinservice/relationship/countLink/15'
```

Request URL

```
http://localhost:8002/coinservice/relationship/countLink/15
```

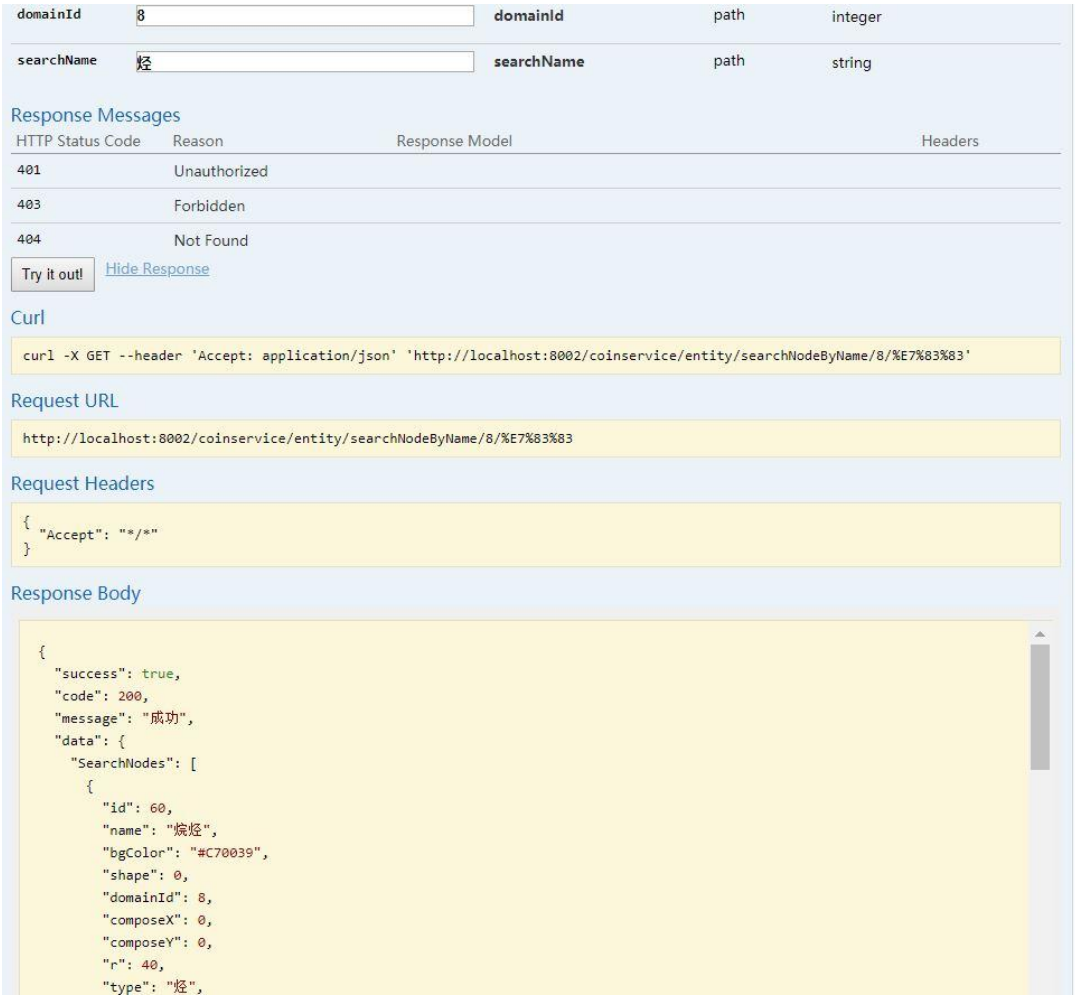
Request Headers

```
{
  "Accept": "*/*"
}
```

Response Body

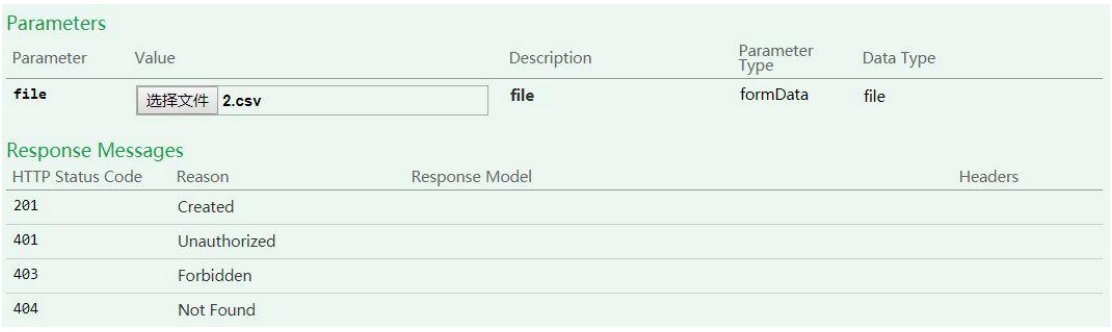

```
{
  "success": true,
  "code": 200,
  "message": "成功",
  "data": {
    "total": 0
  }
}
```

5.4.6 搜索关系



5.5 文件交互测试

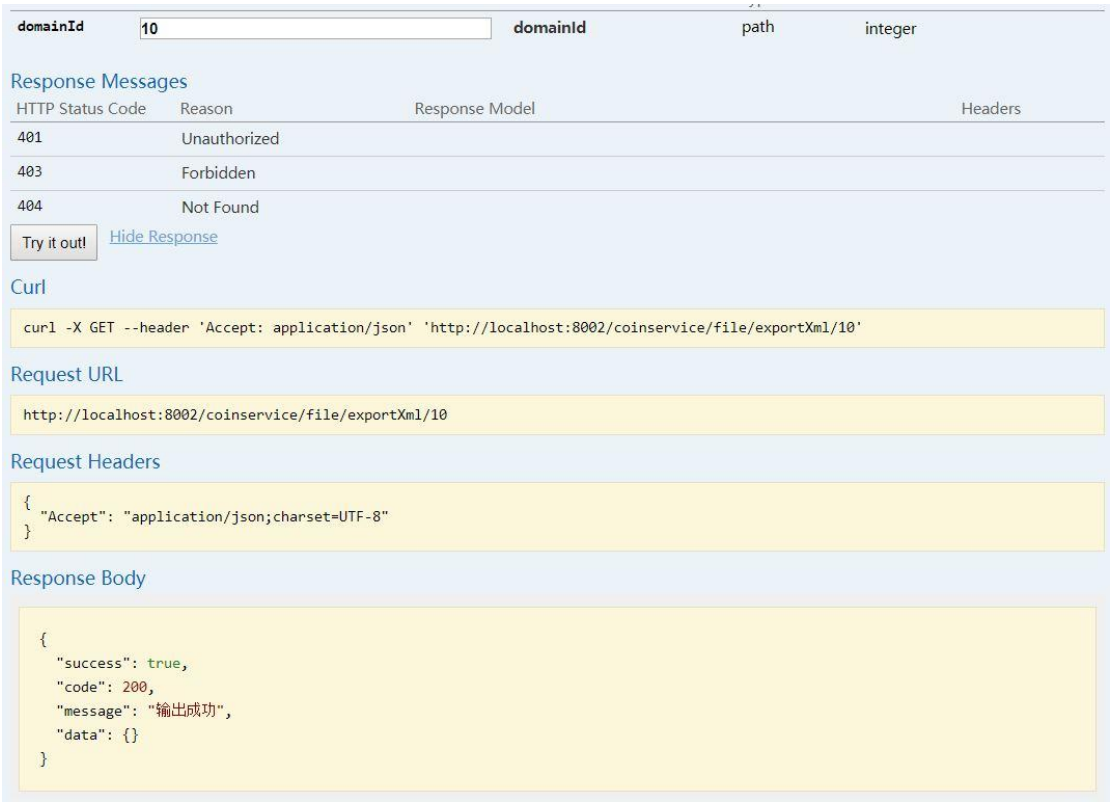
5.5.1 CSV 文件上传生成



Response Body


```
{
  "success": true,
  "code": 200,
  "message": "导入成功",
  "data": {}
}
```


5.5.2 导出 xml 文件



domainId	10	domainId	path	integer

Response Messages

HTTP Status Code	Reason	Response Model	Headers
401	Unauthorized		
403	Forbidden		
404	Not Found		

Try it out! Hide Response

Curl

```
curl -X GET --header 'Accept: application/json' 'http://localhost:8002/coinservice/file/exportXml/10'
```

Request URL

```
http://localhost:8002/coinservice/file/exportXml/10
```

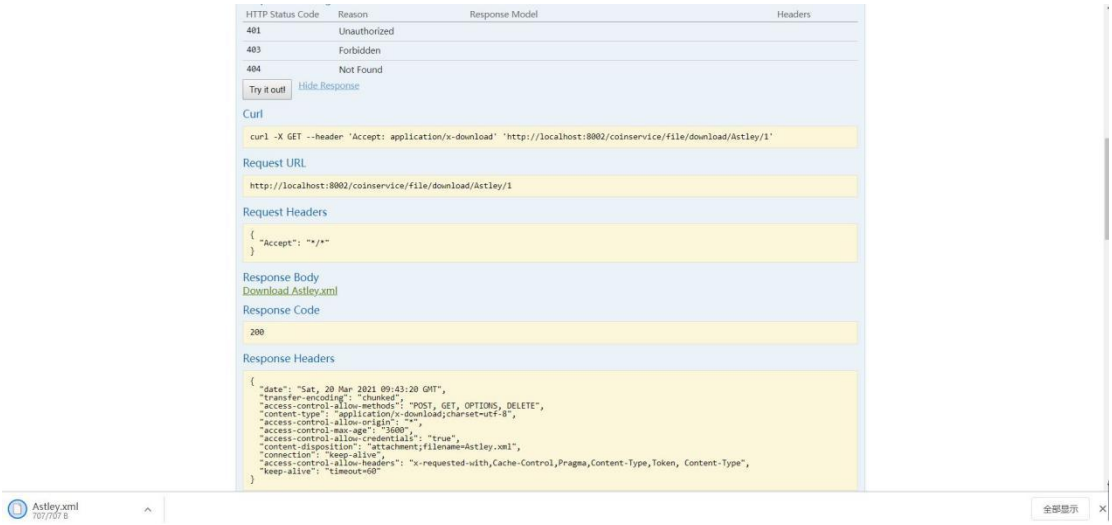
Request Headers


```
{
  "Accept": "application/json;charset=UTF-8"
}
```


Response Body

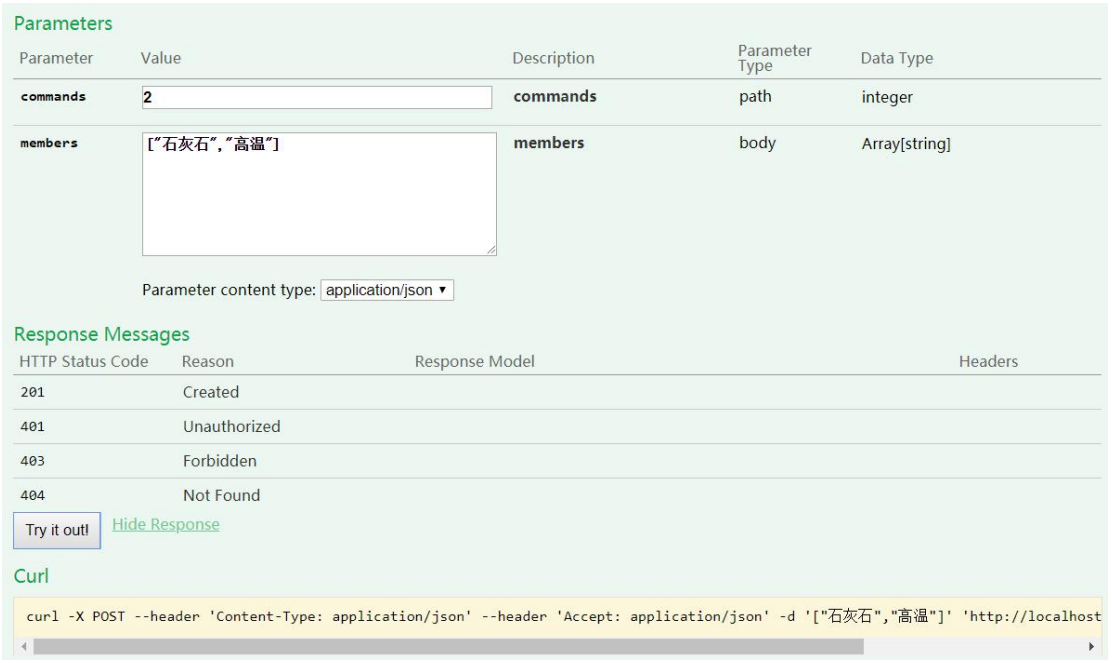

```
{
  "success": true,
  "code": 200,
  "message": "输出成功",
  "data": {}
}
```

5.5.3 下载文件



5.6 文件交互测试

5.6.1 问题文件上传生成



Response Body


```
{
  "success": true,
  "code": 200,
  "message": "成功",
  "data": {
    "result": "氧化钙。"
  }
}
```


6　前端测试报告

6.1 范围

本文档记录了对Coinheap知识图谱可视化系统迭代三前端部分进行测试的过程和结果。

6.2 详细情况

6.2.1 首页功能测试

目标测试文件：index.vue

测试 1：

测试目的：导入 csv 文件生成图谱
输入：导入预定义的 csv 文件选择生成图谱
预期输出：进入工作区，图谱列表更新

测试 2：

测试目的：进入工作区页面跳转
输入：选择“进入工作区”
预期输出：进入目标工作区页面

测试 3：

测试目的：使用推荐模板创建图谱
输入：选择任一推荐模板后选择生成图谱

预期输出：进入工作区，图谱列表更新

目标测试结果：

全部通过

6.2.2 页面顶部标签栏测试

目标测试文件：header.vue

测试 1：

测试目的：标签栏跳转
输入：点击标签栏中前往的目标页面标签
预期输出：进入目标页面

目标测试结果：

全部通过

6.2.3 图谱列表功能测试

目标测试文件：editor.vue

测试 1：

测试目的：创建空白图谱
输入：点击创建图谱选择其中的“创建空白图谱”，输入图谱名称并确认
预期输出：添加成功，图谱列表更新

测试 2：

测试目的：导入 csv 文件创建图谱
输入：点击创建图谱选择其中的“导入 csv 文件”，导入预定义好的 csv 文件

预期输出：添加成功，图谱列表更新

测试 3：

测试目的：使用推荐模板创建图谱 输入：点击创建图谱选择其中的“使用推荐模板”，选择任一推荐模板并确认 预期输出：添加成功，图谱列表更新

测试 4：

测试目的：删除图谱及其中所有内容 输入：选择指定图谱删除 预期输出：弹出确认删除提示框，若选择确认，删除成功，图谱列表更新；若选择取消，提示“已取消删除”

测试 5：

测试目的：选择指定图谱，渲染目标图谱 输入：选择指定图谱 预期输出：若图谱内容不为空，则图谱成功渲染显示

目标测试结果：

全部通过

6.2.4 图谱内部操作测试

目标测试文件：editor.vue

测试 1：

测试目的：右键节点跳出节点操作菜单 输入：选择节点点击右键

预期输出：节点操作菜单显示，菜单包括三个选项：“编辑”、“添加关系”、“删除”

测试 2：

测试目的：编辑节点 输入：选择节点操作菜单中的编辑节点，在表单中对节点各种属性，包括图形、类型、字体大小、半径、描述等进行修改并确认 预期输出：图谱重新渲染，节点更新

测试 3：

测试目的：节点添加关系 输入：选择节点操作菜单中的添加关系，在表单中输入关系名称，选择目标节点并确认 预期输出：表单中无法重新选择起始节点，默认为选中节点；确认后图谱重新渲染，关系添加并渲染成功

测试 4：

测试目的：删除节点 输入：选择节点操作菜单中的删除 预期输出：弹出确认删除提示框，若选择确认，删除成功，图谱重新渲染，指定节点及与其关联的关系都被删除；若选择取消，提示“已取消删除”

测试 5：

测试目的：右键关系跳出关系操作菜单 输入：选择关系点击右键 预期输出：关系操作菜单显示，菜单包括两个选项：“编辑”、“删除”

测试 6：

测试目的：编辑关系
输入：选择关系操作菜单中的编辑，在表单中修改关系名称并确认
预期输出：图谱重新渲染，指定关系名称更新

测试 7：

测试目的：删除关系
输入：选择关系操作菜单中的删除
预期输出：弹出确认删除提示框，若选择确认，删除成功，图谱重新渲染，指定关系消失；若选择取消，提示“已取消删除”

测试 8：

测试目的：拖拽节点
输入：点击节点拖动
预期输出：节点位置跟随鼠标移动位置改变

测试 9：

测试目的：图谱缩放
输入：在图谱区域用鼠标滚轮或触控板进行放大缩小
预期输出：图谱随着鼠标操作放大或缩小

测试 10：

测试目的：图谱布局保存
输入：点击右上角“保存布局”按钮
预期输出：图谱布局保存，下次打开后仍是保存后的布局

测试 11：

测试目的：图谱名称修改 输入：点击“当前图谱”名称右侧的修改按钮，修改图谱名称并回车保存 预期输出：修改成功，图谱列表同步更新

测试 12：

测试目的：鼠标停留节点显示节点信息 输入：鼠标移动至任一节点并停留 1s 以上 预期输出：节点周围出现边框，图谱显示区右侧显示节点详细信息，包括节点名称、节点类型、节点描述

目标测试结果：

全部通过

6.2.5 顶部工具栏功能测试

目标测试文件：editor.vue

测试 1：

测试目的：添加节点按钮 输入：选择工具栏中的添加节点 预期输出：若未选择指定图谱，弹出提示“请先选择要添加节点的图谱哦”；否则弹出添加节点对话框

测试 2：

测试目的：已选中图谱时可以成功添加节点 输入：在添加节点表单中输入节点属性并确认 预期输出：指定图谱重新渲染，被添加节点出现

测试 3：

测试目的：添加关系按钮 输入：选择工具栏中的添加关系 预期输出：若未选择指定图谱，弹出提示“请先选择要添加关系的图谱哦”；否则弹出添加关系对话框

测试 4：

测试目的：已选中图谱时可以成功添加关系 输入：在添加关系表单中输入关系名称，选择起始节点和目标节点并确认 预期输出：指定图谱重新渲染，被添加关系出现

测试 5：

测试目的：导出图片 输入：选择导出功能的导出图片 预期输出：被选中图谱生成图片，浏览器直接下载

测试 6：

测试目的：导出 xml 输入：选择导出功能的导出 xml 预期输出：被选中图谱生成 xml 文件，浏览器直接下载

目标测试结果：

全部通过

6.2.6 可视化能力功能测试

目标测试文件：editor.vue

测试 1：

测试目的：节点搜索
输入：在页面上部的搜索框中输入节点名称或其一部分
预期输出：符合搜索内容的节点被标记，即文本变红加粗，其余节点变透明

测试 2：

测试目的：关系搜索
输入：点击右上角“节点与关系”按钮，点击“关系列表”，在搜索框内输入关系名称或其一部分
预期输出：显示符合条件的关系

测试 3：

测试目的：显示历史搜索
输入：点击搜索框
预期输出：出现以往搜索内容，若无则显示“暂无热门搜索”

测试 4：

测试目的：类型过滤
输入：点击页面上部下拉框，选择需要显示的节点类型
预期输出：没有被选中的类型的所有节点变透明

测试 5：

测试目的：是否显示关系
输入：点击工具栏左侧眼睛状按钮
预期输出：所有关系消失

测试 6：

测试目的：图谱统计功能
输入：无
预期输出：图谱显示区上部实时显示当前图谱中的节点个数和关系个数

测试 7：

测试目的：缩放恢复
输入：点击工具栏左侧恢复按钮
预期输出：图谱恢复到最初大小

测试 8：

测试目的：显示模式切换
输入：来回点击“力导模式”和“排版模式”开关按钮
预期输出：图谱在力导图模式和排版模式之间切换

测试 9：

测试目的：排版模式下布局持久化
输入：进入排版模式，点击“保存布局”按钮
预期输出：布局保存，下次打开仍未保存后的布局

测试 10：

测试目的：“节点与关系”列表显示 输入：选择任一图谱，点击页面右侧的“节点与关系”按钮 预期输出：页面右侧弹出抽屉，包含两个标签页分别显示“节点列表”和“关系列表”；节点列表以卡片形式显示选中图谱的节点样式和相关信息，关系列表以卡片形式显示选中图谱的关系信息

测试 11：

测试目的：节点列表搜索功能 输入：进入“节点与关系”，选择“节点列表”，在上部的搜索框中输入节点名称或其一部分 预期输出：列表中仅显示匹配节点的信息

测试 12：

测试目的：关系列表搜索功能 输入：进入“节点与关系”，选择“关系列表”，在上部的搜索框中输入关系名称或其一部分 预期输出：列表中仅显示匹配关系的信息

测试 13：

测试目的：节点列表按节点类型排序 输入：进入“节点与关系”，选择“节点列表”，选择“按节点类型排序” 预期输出：列表按照节点类型排序显示，相同类型的节点信息连续显示

目标测试结果：

全部通过

6.2.7 右上角用户信息功能测试

目标测试文件：header.vue, index.vue

测试 1：

测试目的：用户头像、昵称显示
输入：无
预期输出：页面右上角在已登录状态下显示用户头像和用户昵称，鼠标移动到该区域后弹出下拉框，显示“个人中心”和“退出”两个选项；在未登录状态下显示登录按钮

测试 2：

测试目的：个人中心跳转
输入：选择用户信息下拉框中的“个人中心”
预期输出：跳转至个人中心页面，显示当前登录用户相关信息

测试 3：

测试目的：用户退出登录
输入：选择用户信息下拉框中的“退出”
预期输出：用户信息清空，直接跳转至登录页面

目标测试结果：

全部通过

6.2.8 登录功能测试

目标测试文件：login.vue

测试 1：

测试目的：正常登录
输入：正确输入已存在用户的手机号和密码，点击登录按钮
预期输出：提示“登录成功”并跳转至首页，右上角显示当前用户信息

测试 2：

测试目的：错误输入提示 输入：错误输入手机号，如输入非 11 位数字；密码字段无输入，点击登录按钮 预期输出：对应字段下方红色字体提示相应错误信息：手机号错误提示“请输入正确格式的手机号”；密码未输入提示“请输入密码”，同时不允许登录

测试 3：

测试目的：登录失败提示 输入：输入已存在用户的手机号但是密码错误；输入不存在用户的手机号和密码 预期输出：显示相应错误提示信息，包括“密码错误”，“该用户不存在”

测试 4：

测试目的：注册用户跳转 输入：选择登录框左下角的“没有账号？立即注册” 预期输出：跳转至注册页面

测试 5：

测试目的：重置密码成功 输入：选择登录框右下角的“忘记密码” 预期输出：弹出重置密码对话框，输入手机号和新密码，选择确认，提示“重置成功”

测试 6：

测试目的：重置密码错误提示 输入：输入的新密码不符合密码格式要求，点击确认 预期输出： 密码字段下方红字提示“密码最少 6 位，至少包括一个字母，一个数字，一个特殊字符”，不允许重置

目标测试结果：

全部通过

6.2.8 注册功能测试

目标测试文件：register.vue

测试 1：

测试目的：正常注册 输入：正确输入 11 位数字格式的手机号及符合格式要求的密码 预期输出：提示“注册成功”并直接登录，提示“登录成功”，跳转至首页，右上角显示当前用户信息

测试 2：

测试目的：错误信息提示 输入：错误输入手机号，如输入非 11 位数字；密码字段不符合格式要求，点击注册按钮 预期输出：对应字段下方红色字体提示相应错误信息：手机号错误提示“请输入正确格式的手机号”；密码不符合格式提示“密码最少 6 位，至少包括一个字母，一个数字，一个特殊字符”，同时不允许注册

测试 3：

测试目的：注册失败 输入：输入已存在用户的手机号 预期输出：提示“该手机号已注册”，不允许重复注册

目标测试结果：

全部通过

6.2.8 个人中心功能测试

目标测试文件：userCenter.vue

测试 1：

测试目的：整体展示 输入： 进入个人中心 预期输出：左侧显示用户图像，若无图像显示“暂无图像”，头像下方显示用户昵称、签名；右侧标签页，分别是“账户信息”、“我的图谱”，若用户为会员，则多一个“会员中心”

测试 2：

测试目的：账号信息显示 输入：选择“账号信息”标签页 预期输出：显示当前登录用户的手机号、用户名、个性签名，下方提供“编辑”、“修改密码”、“注销”三个按钮，若用户非会员，则多一个“充值会员”

测试 3：

测试目的：编辑用户信息 输入：选择“账号信息”标签页，点击“编辑”按钮，修改账户信息，选择确认或取消 预期输出：选择确认，提示“修改成功”，同时更新账户信息；选择取消，回到之前的账户信息展示

测试 4：

测试目的：修改密码 输入：选择“账号信息”标签页，点击“修改密码”按钮，输入原密码和新密码，选择确认 预期输出：若原密码正确，提示“修改成功”；否则提示“原密码错误”，修改失败

测试 5：

测试目的：注销用户 输入：选择“账号信息”标签页，点击“注销”按钮 预期输出：弹出提示“此操作将注销当前账户并清除其所有图谱（不可恢复），是否继续？”，若用户选择确认，则账户被注销，登录信息清空

测试 6：

测试目的：我的图谱显示 输入：选择“我的图谱”标签页 预期输出：列表显示当前用户创建的所有图谱及其创建时间、上次修改时间

测试 7：

测试目的：我的图谱排序 输入：选择“我的图谱”标签页，选择需要排序的字段点击排序箭头 预期输出：列表按照所选字段和排序方式按序显示所有图谱

测试 8：

测试目的：会员中心显示 输入：会员用户选择“会员中心”标签页 预期输出：显示当前用户的会员等级、会员有效期至，及会员权限，下方提供“续费”按钮

测试 9：

测试目的：充值会员及会员续费 输入：选择“账号信息”标签页，点击“充值会员”按钮；或选择“会员中心”标签页，点击“续费”按钮 预期输出：弹出会员充值对话框，选择要充值的时间，点击确认，提示充值成功，同时更新用户信息

测试 10：

测试目的：更换头像 输入：点击头像并选择想要替换为头像的图片 预期输出：鼠标移至头像区域时，头像变灰并显示“更换头像”，点击后弹出文件选择框，选择要替换的头像后确认，提示“修改成功”，并更新用户信息；否则提示错误信息

目标测试结果：

全部通过

6.2.8 问答机器人功能测试

目标测试文件：autoload.js

测试 1：

测试目的：机器人正常功能展示 输入：鼠标移动至机器人区域，或点击其右侧的按钮 预期输出：鼠标移动至机器人区域，机器人弹出对话框，随机显示对话内容；鼠标点击右侧第一个按钮，切换对话内容；点击第三个按钮，切换机器人人物形象；点击第四个按钮，为当前机器人切换服装；点击第五个按钮，隐藏机器人

测试 2：

测试目的：提问 输入：点击机器人右侧第二个问号形状的提问按钮 预期输出：显示提问框，可以在提问框中输入要提问的问题，提问框提供关闭和确认按钮；点击确认后，机器人根据输入的问题显示相应的回答

目标测试结果：

全部通过