测试文档

车一晗 181250009

黄婉红 181840096

纳思彧 181250107

王博 181250133

2021 年 3 月 20 日

摘要

本文档为 Heap 小组在 2021 年春学期《软件工程与计算三》课程作业迭代一中为项目所撰写的测试文档。

目录

1 引言.....4

1.1 编写目的.....4

1.2 对象与范围.....4

1.3 项目背景.....4

1.4 名词与术语.....5

1.5 测试目标.....5

1.6 参考文献.....5

2 测试配套要求.....6

2.1 网络环境.....6

2.2 服务器环境.....6

2.3 测试手段.....6

2.4 测试数据.....6

2.5 测试策略.....6

2.6 测试通过准则.....8

3 单元测试用例.....8

3.1 DomainService 系统.....8

3.1.1 创建工作域模块.....8

3.1.2 删除工作域模块.....8

3.1.3 更新工作域模块.....9

3.1.4 获取指定工作域模块.....9

3.1.5 获取所有工作域模块.....10

3.2 EntityService 系统.....10

3.2.1 创建节点模块.....10

3.2.2 获取指定节点模块.....11

3.2.3 更新节点模块.....11

3.2.4 删除节点模块.....11

3.3 RelationshipService 系统.....12

3.3.1 创建连接模块.....12

3.3.2 更新连接模块.....12

3.3.3 删除连接模块 13
3.3.4 获取指定连接模块 13
3.4 FileService 系统 14
3.4.1 导入 CSV 文件创建图谱模块 14
3.4.2 图谱导出 XML 文件模块 14
4 集成测试用例 15
5 前后端接口交互测试 15
5.1 swagger 说明 15
5.2 DomainController 测试 16
5.2.1 新增域 16
5.1.2 删除域 16
5.2.3 删除域 17
5.2.4 获取域 17
5.3 节点测试 18
5.3.1 新增节点 18
5.3.2 删除节点 18
5.3.3 修改节点 19
5.3.4 显示节点 19
5.4 关系测试 20
5.4.1 新增关系 20
5.4.2 删除关系 21
5.4.3 修改关系 21
5.4.4 显示关系 22
5.5 文件交互测试 22
5.5.1 CSV 文件上传生成 22
5.5.2 导出 xml 文件 23
5.5.3 下载文件 24
6 前端测试报告 24
6.1 范围 24
6.2 详细情况 24

6.2.1 首页功能测试..... 24
6.2.2 页面顶部标签栏测试..... 25
6.2.3 图谱列表功能测试..... 25
6.2.4 图谱内部操作测试..... 26
6.2.5 顶部工具栏功能测试..... 28

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

网络软件：峰值带宽 5Mbps

2.2 服务器环境

服务器硬件：1 核 2G，SSD 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( ));
```

3.2 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);
```

3.3 RelationshipService 系统

3.3.1 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( ));
```

3.3.2 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( ));
```

3.3.3 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);
```

3.3.4 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 FileService 系统

3.4.1 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);
```

3.4.2 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);
```

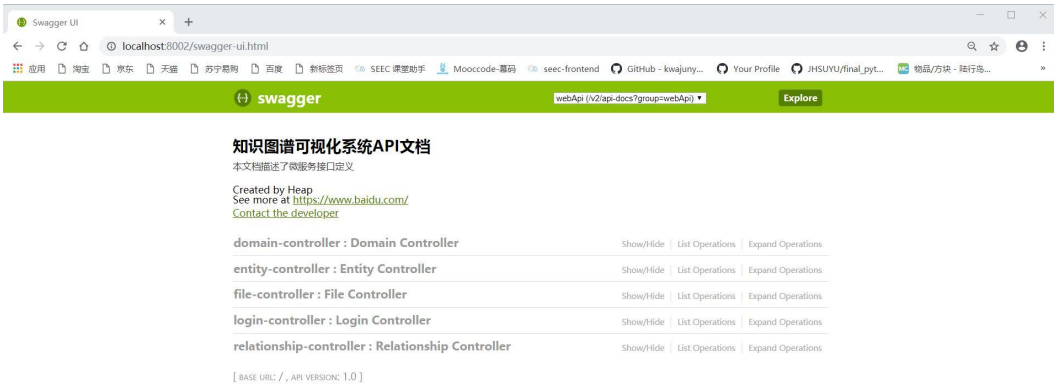
4　集成测试用例

考虑到迭代一较为简单的系统架构设计和集成测试针对复杂系统架构设计的特殊性，该部分集成测试将在迭代二完成。

5　前后端接口交互测试

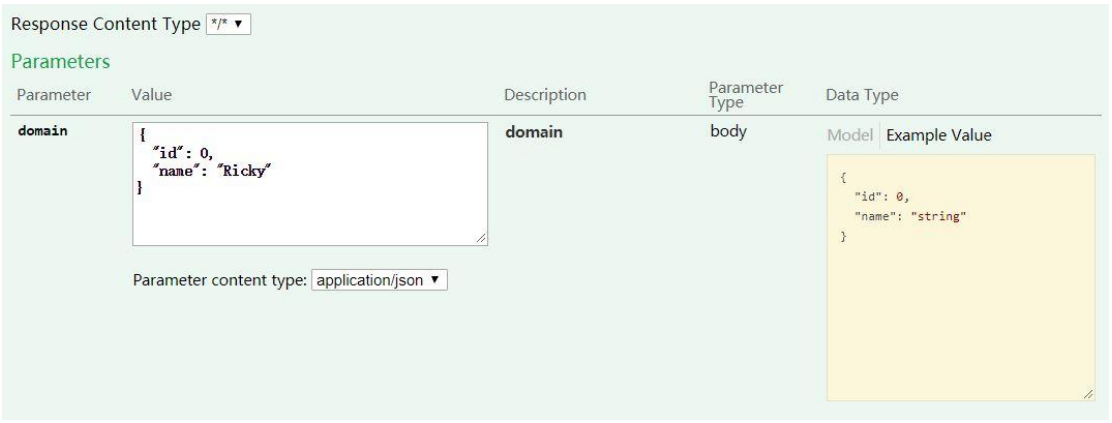
5.1 swagger 说明

通过配置 swagger，可以用于模拟前端向后端的传输接口

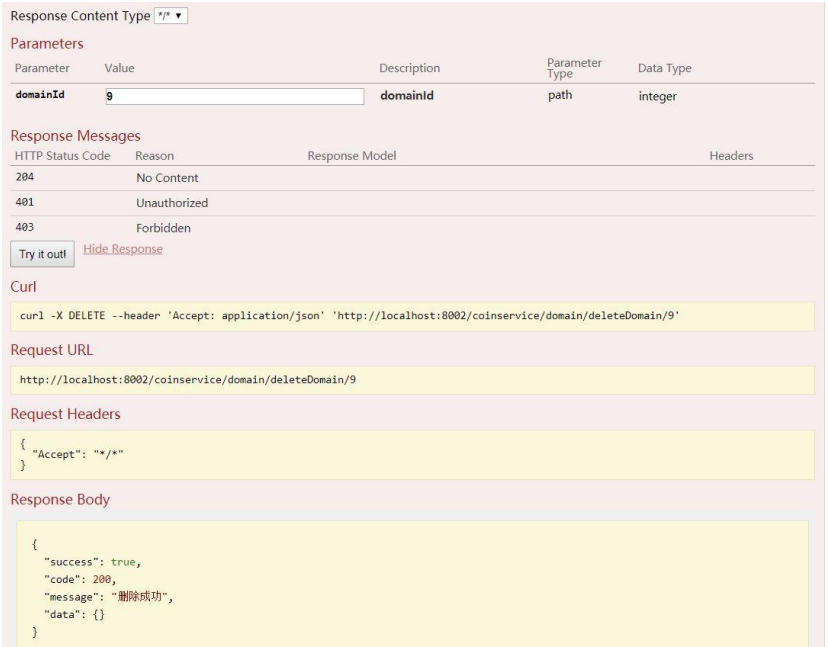


5.2 DomainController 测试

5.2.1 新增域



5.1.2 删除域



5.2.3 修改域

Parameter	Value	Description	Parameter Type	Data Type
domain	{ "id": 10, "name": "Astley" }	domain	body	Model | Example Value

Parameter content type: application/json

```
{
  "id": 0,
  "name": "string"
}
```

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,
```

5.4 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
```

5.5 文件交互测试

5.5.1 CSV 文件上传生成

Parameters

Parameter	Value	Description	Parameter Type	Data Type
file	选择文件 2.csv	**file**	formData	file

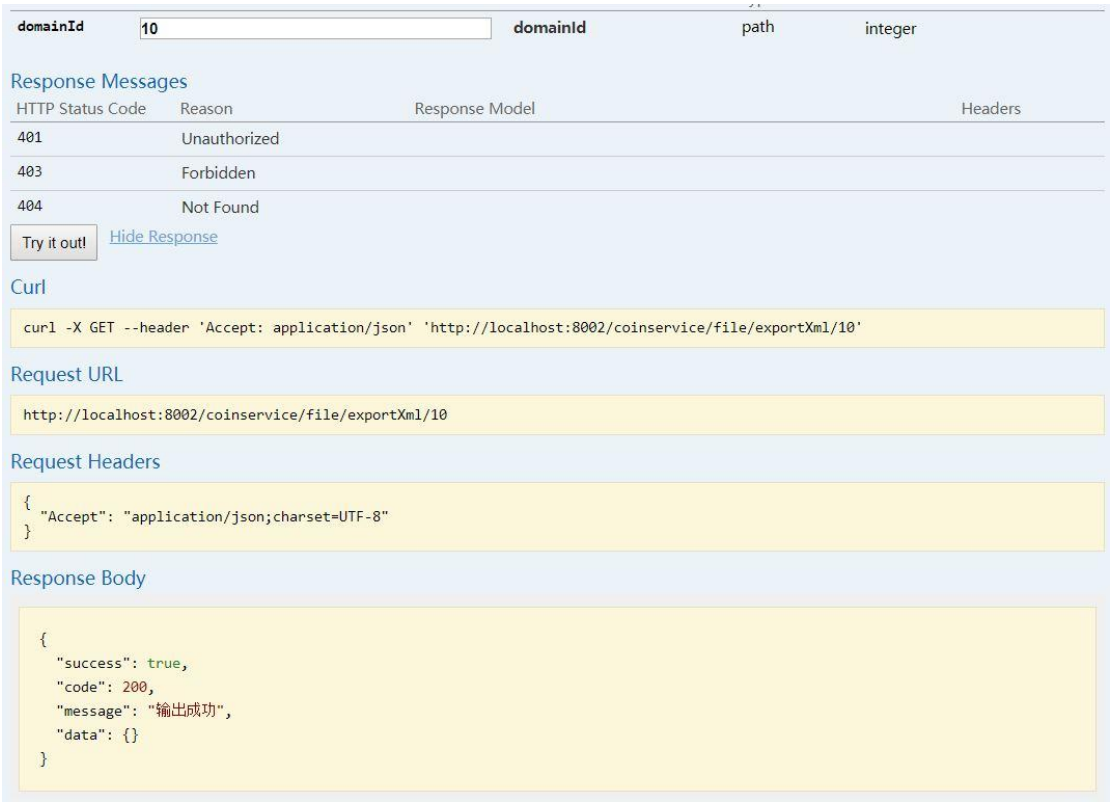
Response Messages

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

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
```

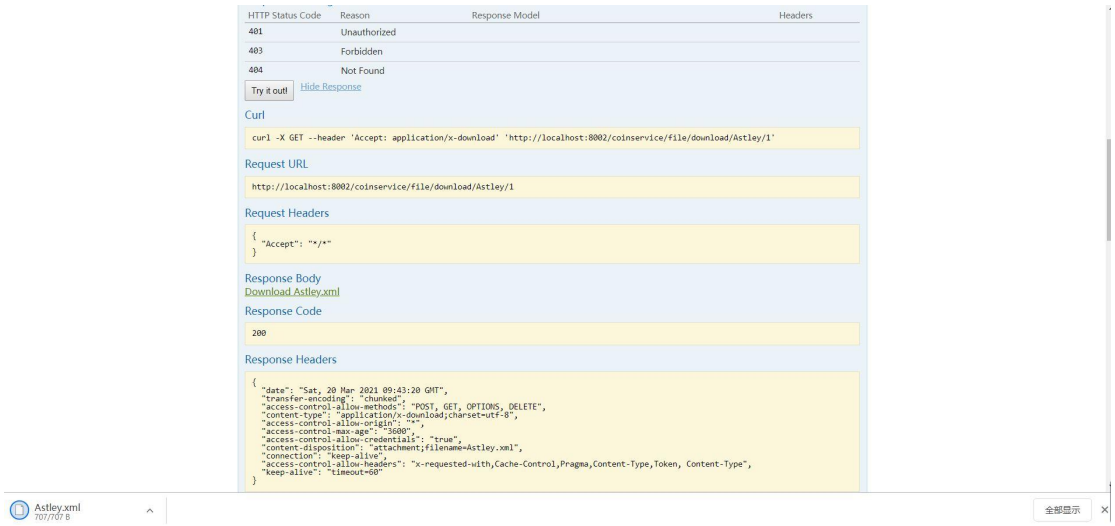
Request Headers


```
{
  "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



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一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

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：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 4：

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，在表单中对节点名称和颜色进行修改并确认 预期输出：图谱重新渲染，节点名称和颜色更新

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

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名称和选择颜色并确认
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