# 项目设计文档

团队名称：Heap

文档更新记录表

|  |  |  |  |
| --- | --- | --- | --- |
| 版本 | 作者 | 版本描述 | 日期 |
| v1.0 | 黄婉红 | 初稿模板 | 2021-03-05 |
|  |  |  |  |

目录

[项目设计文档 1](#_Toc65935417)

[1. 引言 2](#_Toc65935418)

[1.1 编写目的 2](#_Toc65935419)

[1.2 对象与范围 2](#_Toc65935420)

[1.3 参考文献 2](#_Toc65935421)

[1.4 名词与术语 2](#_Toc65935422)

[2. 逻辑视角 2](#_Toc65935423)

[2.1 分层架构图 2](#_Toc65935424)

[2.2 逻辑包图 2](#_Toc65935425)

[3. 组合视角 2](#_Toc65935426)

[3.1 物理包划分 2](#_Toc65935427)

[开发包 2](#_Toc65935428)

[依赖的其他开发包 2](#_Toc65935429)

[3.2 物理包图 3](#_Toc65935430)

[4. 接口视角 3](#_Toc65935431)

[4.1 模块的职责 3](#_Toc65935432)

[4.2 模块的接口规范 3](#_Toc65935433)

[5. 信息视角 4](#_Toc65935434)

[5.1 VO定义 4](#_Toc65935435)

[含义 4](#_Toc65935436)

[属性 4](#_Toc65935437)

[字段 4](#_Toc65935438)

## 1. 引言

### 1.1 编写目的

本文档提供COIN知识图谱系统的软件架构概览，采用若干架构视图描述系统的不同方面，以便表示构造系统所需要的重要架构决策

### 1.2 对象与范围

本文档的读者是Heap团队内部的开发和管理人员，参考了RUP的《软件架构文档模板》，用于指导下一循环的代码开发和测试工作

### 1.3 参考文献

《软件需求规格说明书》, Heap;

《软件架构文档模板》, Rational Software Corporation;

### 1.4 名词与术语

COIN：COIN知识图谱系统

## 2. 逻辑视角

### 2.1 分层架构图

### 2.2 逻辑包图

## 3. 组合视角

### 3.1 物理包划分

|  |  |
| --- | --- |
| 开发包 | 依赖的其他开发包 |
|  |  |
|  |  |
|  |  |

### 3.2 物理包图

## 4. 接口视角

### 4.1 模块的职责

学长文档这块空着？

### 4.2 模块的接口规范

#### 4.2.1 用户界面层分解（待定）

|  |  |
| --- | --- |
| 模块 | 职责 |
|  |  |
| 需要的服务（需接口） | |
| 服务名 | 服务 |
|  |  |

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
|  | 语法 |  |
| 前置条件 |  |
| 后置条件 |  |

#### 4.2.2 业务逻辑层分解

**4.2.2.1 EntityService的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| 模块 | | 职责 | |
| EntityService | | 负责对节点的职责 | |
| 需要的服务（需接口） | | | |
| 服务名 | | 服务 | |
| EntityService.createNode | | 创建新节点 | |
| EntityService.updateNode | | 更新节点信息 | |
| EntityService.deleteNode | | 删除现有节点 | |
| EntityService.getNodeByDomainId | | 查找现有域内的所有节点 | |
| 提供的服务（供接口） | | | | |
| EntityService.createNode | 语法 | | Entity createNode(String name, String color, int type, int domainId); | |
| 前置条件 | | DomainId和name不为空 | |
| 后置条件 | | 根据输入，创建节点信息 | |
| EntityService.updateNode | 语法 | | Entity updateNode(Entity entity); | |
| 前置条件 | | 被更新的节点已存在 | |
| 后置条件 | | 根据输入，修改节点信息 | |
| EntityService.deleteNode | 语法 | | boolean deleteNode(Entity entity); | |
| 前置条件 | | 被更新的节点已存在 | |
| 后置条件 | | 根据输入，删除该节点 | |
| EntityService.getNodeByDomainId | 语法 | | List<Entity> getNodeByDomainId(int domainId); | |
| 前置条件 | | 输入域名合法 | |
| 后置条件 | | 根据输入，返回所有节点信息 | |

**4.2.2.2 RelationshipService的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| 模块 | | 职责 | |
| RelationshipService | | 负责对节点间关系的职责 | |
| 需要的服务（需接口） | | | |
| 服务名 | | 服务 | |
| RelationshipService.createLink | | 创建新节点关系 | |
| RelationshipService.updateLink | | 更新关系信息 | |
| RelationshipService.deleteLink | | 删除现有关系 | |
| RelationshipService.getLinkByDomainId | | 查找现有域内的所有关系图谱 | |
| 提供的服务（供接口） | | | | |
| RelationshipService.createLink | 语法 | | Relationship createLink(Long fromId, Long toId, String name); | |
| 前置条件 | | FromId和toId不为空且存在 | |
| 后置条件 | | 根据输入，创建两节点间关系 | |
| RelationshipService.updateLink | 语法 | | Relationship updateLink(Relationship relationship); | |
| 前置条件 | | 被更新的关系已存在 | |
| 后置条件 | | 根据输入，修改关系信息 | |
| RelationshipService.deleteLink | 语法 | | boolean deleteLink(Relationship relationship); | |
| 前置条件 | | 被更新的关系已存在 | |
| 后置条件 | | 根据输入，删除该关系 | |
| RelationshipService.getLinkByDomainId | 语法 | | List<Relationship> getLinkByDomainId(int domainId); | |
| 前置条件 | | 输入域名合法 | |
| 后置条件 | | 根据输入，返回域内所有关系图谱（含孤立节点） | |

#### 4.2.3 数据层分解

|  |  |  |  |
| --- | --- | --- | --- |
| 模块 | | | 职责 |
| EntityMapper | | | 负责对节点数据的访问控制 |
| RelationshipMapper | | | 负责对关系数据的访问控制 |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| EntityMapper.getNodeByDomainId | | | 根据域名返回所有节点 |
| EntityMapper.updateNode | | | 更新节点信息 |
| EntityMapper.getUnLinkNodes | | | 返回域名内的无关系孤立节点 |
| RelationshipMapper.updateLink | | | 更新关系信息 |
| RelationshipMapper.getLinkByDomainId | | | 根据域名返回所有关系（不含孤立节点） |
| Neo4jRepository.save | | | 新增并保存关系/节点 |
| Neo4jRepository.delete | | | 删除现有关系/节点 |
| 提供的服务（供接口） | | | | |
| EntityMapper.updateNode | 语法 | Entity updateNode(@Param("id") Long id,@Param("name") String name,@Param("bgColor") String bgColor); | | |
| 前置条件 | 节点id不为空且存在 | | |
| 后置条件 | 根据其他输入，更新节点内容 | | |
| EntityMapper.getNodeByDomainId | 语法 | List<Entity> getNodeByDomainId(@Param("domainId") int domainId); | | |
| 前置条件 | 域名domainid不为空且存在 | | |
| 后置条件 | 根据输入，返回该域内所有节点 | | |
| EntityMapper.getUnLinkNodes | 语法 | List<Entity> getUnLinkNodes(@Param("domainId") int domainId); | | |
| 前置条件 | 域名domainid不为空且存在 | | |
| 后置条件 | 根据输入，返回该域内所有孤立节点 | | |
| RelationShipMapper.updateLink | 语法 | Relationship updateLink(@Param("id") Long id,@Param("name") String name); | | |
| 前置条件 | 关系id不为空且存在 | | |
| 后置条件 | 根据其他输入，更新关系内容 | | |
| RelationShipMapper.getLinkByDomainId | 语法 | List<Relationship> getLinkByDomainId(@Param("domainId") int domainId); | | |
| 前置条件 | 域名domainid不为空且存在 | | |
| 后置条件 | 根据输入，返回该域内所有关系 | | |

## 5. 信息视角

### 5.1 VO定义

|  |  |  |
| --- | --- | --- |
| 含义 | 属性 | 字段 |
|  |  |  |