**题库系统详细设计**

# 关键词定义

## 文档（doc）

系统中的对象均称为文档，如知识体系，题目，视频等。已知文档如下：

|  |  |
| --- | --- |
| 文档名称 | 文档类型 |
| 题目 | 素材 |
| 视频 | 素材 |
| 知识素材 | 素材 |
| 讲义 | 组件 |
| 试卷 | 组件 |

## 标签（tag）

文档的搜索条件，可以灵活的添加到文档中，如学科，学部，知识点等。

### 结构化标签

结构确定的标签，如学科，学部，知识点等，使用精确匹配（id）进行搜索，分为一层标签（如学科，学部）和树形标签（如知识点，来源）两类，两类标签都可以进行多对一关联。

### 非结构化标签

结构不确定的标签，如文本标签，使用模糊匹配（文本）进行搜索。

### 已知标签

|  |  |  |
| --- | --- | --- |
| 名称 | 标签类型 | 子类型 |
| 学科 | 结构化标签 | 一层标签 |
| 学部 | 结构化标签 | 一层标签 |
| 范围 | 结构化标签 | 一层标签 |
| 难度 | 结构化标签 | 一层标签 |
| 年份 | 结构化标签 | 一层标签 |
| 层次 | 结构化标签 | 一层标签 |
| 来源 | 结构化标签 | 树形标签 |
| 知识点 | 结构化标签 | 树形标签 |
| 文本标签 | 非结构化标签 |  |

## 标签的关联（tag-relation）

两个不同的标签之间会存在关联关系，需要进行关联，已知关联如下：

|  |  |  |
| --- | --- | --- |
| 上级标签 | 下级标签 | 关联关系 |
| 学科 | 学部 | 多对多 |

## 文档和标签的关联（doc-tag-relation）

不同的文档可使用的标签也不相同，需要进行关联，已知关联如下：

|  |  |  |
| --- | --- | --- |
| 文档 | 标签 | 关联关系 |
| 题目 | 学科 | 有且只有一个标签 |
| 题目 | 学部 | 有且只有一个标签 |
| 题目 | 范围 | 有且只有一个标签 |
| 题目 | 难度 | 有且只有一个标签 |
| 题目 | 年份 | 有且只有一个标签 |
| 题目 | 层次 | 有，可有多个标签，不区分主副 |
| 题目 | 来源 | 有，可有多个标签，不区分主副 |
| 题目 | 知识点 | 有，可有多个标签，区分主副 |
| 题目 | 文本标签 | 可无，可有多个标签，不区分主副 |
| 视频 | 学科 | 有且只有一个标签 |
| 视频 | 学部 | 有且只有一个标签 |
| 视频 | 年份 | 有且只有一个标签 |
| 视频 | 知识点 | 有，可有多个标签，区分主副 |
| 视频 | 文本标签 | 可无，可有多个标签，不区分主副 |
| 知识素材 | 学科 | 有且只有一个标签 |
| 知识素材 | 学部 | 有且只有一个标签 |
| 知识素材 | 年份 | 有且只有一个标签 |
| 知识素材 | 知识点 | 有，可有多个标签，区分主副 |
| 知识素材 | 文本标签 | 可无，可有多个标签，不区分主副 |
| 试卷 | 学科 | 有且只有一个标签 |
| 试卷 | 学部 | 有且只有一个标签 |
| 试卷 | 年份 | 有且只有一个标签 |
| 试卷 | 知识点 | 有，可有多个标签，区分主副 |
| 试卷 | 难度 | 有且只有一个标签 |
| 试卷 | 范围 | 有且只有一个标签 |
| 试卷 | 文本标签 | 可无，可有多个标签，不区分主副 |
| 讲义 | 学科 | 有且只有一个标签 |
| 讲义 | 学部 | 有且只有一个标签 |
| 讲义 | 年份 | 有且只有一个标签 |
| 讲义 | 知识点 | 有，可有多个标签，区分主副 |
| 讲义 | 难度 | 有且只有一个标签 |
| 讲义 | 范围 | 有且只有一个标签 |
| 讲义 | 文本标签 | 可无，可有多个标签，不区分主副 |

# 数据库设计

## 标签数据库设计

### 标签类型表（mysql）

名称：tag\_type

|  |  |  |
| --- | --- | --- |
| 字段名称 | 类型 | 说明 |
| Id | Bigint | 主键 |
| name | Varchar(50) | 标签类型名称 |
| classify | tinyint | 标签分类，值（0：一层标签，1：来源，2：主知识点，3：知识点，4：文本） |
| type | tinyint | 标签类型，值（0：系统，1：自定义） |
| search\_key | Varchar(50) | 搜索使用的key，只使用英文和数字 |
| create\_time | timestamp | 创建时间 |
| update\_time | timestamp | 最近更新时间 |

### 一层标签值表（mysql）

名称：tag\_item

|  |  |  |
| --- | --- | --- |
| 字段名称 | 类型 | 说明 |
| Id | Bigint | 主键 |
| tag\_type\_id | Bigint | 对应标签类型主键 |
| name | Varchar(100) | 标签值 |
| create\_time | timestamp | 创建时间 |
| update\_time | timestamp | 最近更新时间 |

### 一层标签关系表（mysql）

名称：tag\_rel

说明：表示两个标签之间的关联关系，一条记录表示当选择了tag\_type的tag\_item时，可以选择关联relation\_type的relation\_item

|  |  |  |
| --- | --- | --- |
| 字段名称 | 类型 | 说明 |
| Id | Bigint | 主键 |
| tag\_type\_id | Bigint | 对应标签类型表主键 |
| tag\_item\_id | Bigint | 对应标签值表主键 |
| relation\_type\_id | Bigint | 对应标签类型表主键 |
| relation\_item\_id | Bigint | 对应标签值表主键 |
| create\_time | timestamp | 创建时间 |
| update\_time | timestamp | 最近更新时间 |

### 来源表（mysql）

名称：source

说明：来源是一类树形标签

|  |  |  |
| --- | --- | --- |
| 字段名称 | 类型 | 说明 |
| Id | Bigint | 主键 |
| name | Varchar(50) | 来源名称 |
| type | Varchar(100) | 来源类型，表示使用该来源的对象，值为该对象的mysql表名称 |
| create\_time | timestamp | 创建时间 |
| update\_time | timestamp | 最近更新时间 |

### 来源标签表（mysql）

名称：source\_item

说明：支持用户自定义标签层次和值，预留5个层次

|  |  |  |
| --- | --- | --- |
| 字段名称 | 类型 | 说明 |
| Id | Bigint | 主键 |
| source\_id | Bigint | 对应来源主键 |
| tag\_type1 | Varchar(50) | 第一层标签名称 |
| tag\_item1 | Varchar(100) | 第一层标签值 |
| tag\_type2 | Varchar(50) | 第二层标签名称 |
| tag\_item2 | Varchar(100) | 第二层标签值 |
| tag\_type3 | Varchar(50) | 第三层标签名称 |
| tag\_item3 | Varchar(100) | 第三层标签值 |
| tag\_type4 | Varchar(50) | 第四层标签名称 |
| tag\_item4 | Varchar(100) | 第四层标签值 |
| tag\_type5 | Varchar(50) | 第五层标签名称 |
| tag\_item5 | Varchar(100) | 第五层标签值 |
| create\_time | timestamp | 创建时间 |
| update\_time | timestamp | 最近更新时间 |

### 文本标签表（mysql）

名称：text\_item

|  |  |  |
| --- | --- | --- |
| 字段名称 | 类型 | 说明 |
| Id | Bigint | 主键 |
| name | Varchar(100) | 文本标签名称 |
| type | Varchar(50) | 文本标签类型，表示使用该标签的文档，值为该对象的mysql表名称 |
| deleted | tinyint | 是否删除，值为（0：是，1：否） |
| user\_id | Bigint | 用户id，本期不做功能 |
| create\_time | timestamp | 创建时间 |
| update\_time | timestamp | 最近更新时间 |

### 文档类型和一层标签类型对应表（mysql）本期不做

名称：doc\_tagtype\_rel

|  |  |  |
| --- | --- | --- |
| 字段名称 | 类型 | 说明 |
| Id | Bigint | 主键 |
| doc\_type | Varchar(100) | 文档类型，值为该文档的mysql表名称 |
| tag\_type\_id | Bigint | 对应一层标签类型表主键 |
| relation\_type | Varchar(20) | 文档和一层标签的关系类型，值（一对一，一对多） |
| create\_time | timestamp | 创建时间 |
| update\_time | timestamp | 最近更新时间 |

### 文档和标签值对应表（mysql）

名称：doc\_tagitem\_rel

说明：这里统一定义，可按文档分别建表，拆分建表后要去掉doc\_type字段

|  |  |  |
| --- | --- | --- |
| 字段名称 | 类型 | 说明 |
| Id | Bigint | 主键 |
| doc\_type | Varchar(100) | 文档类型，值为该文档的mysql表名称（如果按文档拆分建表，需去掉该字段） |
| doc\_id | Bigint | 文档id |
| tag\_type\_id | Bigint | 对应标签类型表主键 |
| tag\_item\_id | Bigint | 对应标签值表主键 |
| weight | int | 权重，表示一对多标签中的主副关系，（主：20，副：10） |
| create\_time | timestamp | 创建时间 |
| update\_time | timestamp | 最近更新时间 |

## 知识体系数据库设计

### 主知识体系表（mysql）

名称：knowledge\_hierarchy\_main

|  |  |  |
| --- | --- | --- |
| 字段名称 | 类型 | 说明 |
| Id | Bigint | 主键 |
| name | Varchar(100) | 知识体系名称（目前好像不用，保留字段） |
| subject\_id | Bigint | 对应学科主键（学科是系统的一层标签） |
| scope\_id | Bigint | 对应范围主键（范围是系统的一层标签） |
| department\_id | Bigint | 对应学部主键（学部是系统的一层标签） |
| create\_time | timestamp | 创建时间 |
| update\_time | timestamp | 最近更新时间 |

### 主知识点表（mysql）

名称：knowledge\_point\_main

|  |  |  |
| --- | --- | --- |
| 字段名称 | 类型 | 说明 |
| Id | Bigint | 主键 |
| hierarchy\_id | Bigint | 对应主知识体系id |
| parent\_id | Bigint | 表示树形体系的父节点id，第一层对应为0，其余对应本表id |
| name | Varchar(100) | 知识点名称 |
| level | int | 表示在树中的深度，从1开始递增 |
| path | Varchar(255) | 表示到达该知识点的路径，以上级知识点id拼接，以“/”分割，第一层为空 |
| is\_leaf | tinyint | 是否是叶节点，值（0:是，1:否） |
| create\_time | timestamp | 创建时间 |
| update\_time | timestamp | 最近更新时间 |

### 知识体系表（mysql）

名称：knowledge\_hierarchy

|  |  |  |
| --- | --- | --- |
| 字段名称 | 类型 | 说明 |
| Id | Bigint | 主键 |
| name | Varchar(100) | 知识体系名称 |
| main\_id | Bigint | 对应主知识体系id |
| subject\_id | Bigint | 对应学科主键 |
| scope\_id | Bigint | 对应范围主键 |
| department\_id | Bigint | 对应学部主键 |
| create\_time | timestamp | 创建时间 |
| update\_time | timestamp | 最近更新时间 |

### 知识点表（mysql）

名称：knowledge\_point

说明：一个知识体系是对应主知识体系的子树

|  |  |  |
| --- | --- | --- |
| 字段名称 | 类型 | 说明 |
| Id | Bigint | 主键 |
| hierarchy\_id | Bigint | 对应知识体系id |
| main\_point\_id | Bigint | 对应主知识点id |
| is\_use | tinyint | 是否引用，值为（0：是，1：否） |
| create\_time | timestamp | 创建时间 |
| update\_time | timestamp | 最近更新时间 |

## 素材数据库设计

### 题目表（mysql）

名称：question

|  |  |  |
| --- | --- | --- |
| 字段名称 | 类型 | 说明 |
| Id | Bigint | 主键 |
| derive\_type | tinyint | 衍生类型，值（0：母题，1：衍生题） |
| derive\_id | Bigint | 母题id |
| status | tinyint | 审核状态，值（0：内容未确认，1：未打标签，2：待校验，3：待审核，4：通过，5：待修改） |
| create\_time | timestamp | 创建时间 |
| update\_time | timestamp | 最近更新时间 |

### 题目版本表（mysql+mongodb）

Mysql表名称：question\_version

|  |  |  |
| --- | --- | --- |
| 字段名称 | 类型 | 说明 |
| Id | Bigint | 主键 |
| main\_id | Bigint | 对应题目表主键 |
| mongo\_id | Varchar(100) | 对应的Mongodb主键 |
| version | Varchar(100) | 版本号（生成规则待定） |
| is\_new | tinyint | 是否是最新版本，值（0：是，1：否） |
| type | tinyint | 题型，本期只支持单选，多选，填空，解答，值依次为0,1,2,3 |
| create\_time | timestamp | 创建时间 |
| update\_time | timestamp | 最近更新时间 |

Mongodb集合名称：question

|  |  |  |
| --- | --- | --- |
| 字段名称 | 类型 | 说明 |
| Id | Object\_id | 主键 |
| mysql\_id | long | 对应mysql题目表主键 |
| version\_id | long | 对应mysql题目版本表主键 |
| content | json | 题干内容 |
| question | json | 提问内容（选择题的选项，填空题的空白，问答题的提问） |
| result | json | 答案内容 |
| analysis | json | 解析内容 |
| text\_label | json | 文本标签 |

### 视频表（mysql）

名称：video

|  |  |  |
| --- | --- | --- |
| 字段名称 | 类型 | 说明 |
| Id | Bigint | 主键 |
| create\_time | timestamp | 创建时间 |
| update\_time | timestamp | 最近更新时间 |

### 视频版本表（mysql+mongodb）

Mysql表名称：video\_version

|  |  |  |
| --- | --- | --- |
| 字段名称 | 类型 | 说明 |
| Id | Bigint | 主键 |
| main\_id | Bigint | 对应视频表主键 |
| mongo\_id | Varchar(100) | 对应的Mongodb主键 |
| version | Varchar(100) | 版本号（生成规则待定） |
| is\_new | tinyint | 是否是最新版本，值（0：是，1：否） |
| type | tinyint | 题型，本期只支持单选，多选，填空，解答，值依次为0,1,2,3 |
| name | Varchar(100) | 视频名称 |
| speaker | Varchar(50) | 主讲人 |
| url | Varchar(255) | 视频url |
| create\_time | timestamp | 创建时间 |
| update\_time | timestamp | 最近更新时间 |

Mongodb集合名称：video

|  |  |  |
| --- | --- | --- |
| 字段名称 | 类型 | 说明 |
| Id | Object\_id | 主键 |
| mysql\_id | long | 对应mysql视频表主键 |
| version\_id | long | 对应mysql视频版本表主键 |
| info | json | 视频信息 |
| text\_label | json | 文本标签 |

### 知识素材表（mysql）

名称：knowledge

|  |  |  |
| --- | --- | --- |
| 字段名称 | 类型 | 说明 |
| Id | Bigint | 主键 |
| create\_time | timestamp | 创建时间 |
| update\_time | timestamp | 最近更新时间 |

### 知识素材版本表（mysql+mongodb）

Mysql表名称：knowledge\_version

|  |  |  |
| --- | --- | --- |
| 字段名称 | 类型 | 说明 |
| Id | Bigint | 主键 |
| main\_id | Bigint | 对应知识素材表主键 |
| mongo\_id | Varchar(100) | 对应的Mongodb主键 |
| version | Varchar(100) | 版本号（生成规则待定） |
| is\_new | tinyint | 是否是最新版本，值（0：是，1：否） |
| name | Varchar(100) | 知识素材名称 |
| create\_time | timestamp | 创建时间 |
| update\_time | timestamp | 最近更新时间 |

Mongodb集合名称：knowledge

|  |  |  |
| --- | --- | --- |
| 字段名称 | 类型 | 说明 |
| Id | Object\_id | 主键 |
| mysql\_id | long | 对应mysql知识素材表主键 |
| version\_id | long | 对应mysql知识素材版本表主键 |
| content | json | 知识素材正文 |
| text\_label | json | 文本标签 |

## 组件数据库设计

### 试卷表（mysql+mongodb）

试卷mysql表设计：test\_paper

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 说明 |
| id | bigint | 主键 |
| name | varchar(50) | 组件名称 |
| main\_id | bigint | 试卷id（首次添加时生成，规则待定） |
| mongo\_id | varchar(100) | 数据id，mongodb中存储（卷子的内容） |
| folder\_id | bigint | 组件文件夹id |
| version | varchar(100) | 版本号 |
| is\_new | tinyint | 是否最新的版本 |
| deleted | tinyint | 是否已删除 |
| difficulty | tinyint | 难度系数 |
| create\_ime | timestamp | 创建时间 |
| create\_user | bigint | 创建人id |
| update\_time | timestamp | 最后修改时间 |
| update\_user | bigint | 最后修改人id |

试卷mongodb集合设计：test\_paper

|  |
| --- |
| {  "\_id" : objectid("5b32f9b40c37c5b307ec7967"),  “question\_num”:”题量”  "show\_knowledge\_point":是否显示知识点  “show\_difficulty”：是否显示难度  “show\_source”：是否显示来源  "structure" : [  {      “question”:[  {  "number" : "题号",  “index”:”索引”,  “type”:” 类型：标题，题目”  “title\_level”:”几级标题”  }  {  " number " : "题号",  “index”:”索引”,  “type”:” 类型：标题，题目”  "question\_id" : "试题id",  "score" : "分数",  “answer”:”答案”,  “analysis”:”解析”  },  {  " number " : "题号",  “index”:”索引”,  “type”:” 类型：标题，题目”  "question\_id" : "试题id",  "score" : "分数",  “answer”:”答案”，  “analysis”:”解析”  }  ]  },  ],  "answers" : [  {  "question\_id" : "试题id",  "answer" : "答案a"  },  {  "question\_id" : "试题id",  "answer" : "答案b"  }  ]  } |

### 讲义表（mysql+mongodb）

讲义mysql表设计：handout \_paper

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 说明 |
| id | bigint | 主键 |
| name | varchar(50) | 组件名称 |
| main\_id | bigint | 讲义id（首次添加时生成，规则待定） |
| mongo\_id | varchar(100) | 数据id，mongodb中存储（讲义的结构） |
| folder\_id | bigint | 组件文件夹id |
| version | varchar(100) | 版本号 |
| is\_new | tinyint | 是否最新版本 |
| deleted | tinyint | 是否已删除 |
| create\_time | timestamp | 创建时间 |
| create\_user | bigint | 创建人id |
| update\_time | timestamp | 最后修改时间 |
| update\_user | bigint | 最后修改人id |

讲义mongodb集合设计：handout \_paper

|  |
| --- |
| {  "\_id" : objectid("5b32f9b40c37c5b307ec7967"),  "structure" : [  {  "order" : "素材序号",  "type" : "类型：（知识点素材）",  “content”:[  {  "order" : "序号",  "knowledge\_id" : "知识点素材id",  "show\_score" : "是否显示分数",  "show\_difficulty" : "是否显示难度",  "show\_knowledge\_point" : "是否显示知识点"  }  ]  },  {  "order" : "题号",  "type" : "类型：（选择题，填空题，判断题等）",  “content”:[  {  "order" : "序号",  "question\_id" : "mysqlid",  "score" : "分数",  “answer”:”答案”,  “analysis”:”解析”，  "show\_score" : "是否显示分数",  "show\_answer" : "是否显示答案",  "show\_difficulty" : "是否显示难度",  "show\_knowledge\_point" : "是否显示知识点",  "show\_analysis" : "是否显示解析"  },  {  "order" : "序号",  "question\_id" : "mysqlid",  "score" : "分数",  “answer”:”答案”,  “analysis”:”解析”  "show\_score" : "是否显示分数",  "show\_answer" : "是否显示答案",  "show\_difficulty" : "是否显示难度",  "show\_knowledge\_point" : "是否显示知识点",  "show\_analysis" : "是否显示解析"  }  ]  }  ],  "answers" : [  {  "question\_id" : "试题id",  "answer" : "答案a"  },  {  "question\_id" : "试题id",  "answer" : "答案b"  }  ]  } |

### 组件和试题解析的关系表（module\_analysis\_rel）

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 说明 |
| id | bigint | 主键 |
| paper\_id | bigint | 主键 |
| paper\_type | varchar(50) | 试卷讲义表 |
| question\_id | bigint | 试题id |
| analysis | bigint | 解析id |
| create\_time | timestamp | 创建时间 |
| update\_time | timestamp | 最后修改时间 |

### 组件和素材关系表 (module\_material\_rel)

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 说明 |
| id | bigint | 主键 |
| module\_id | bigint | 组件id |
| module\_type | varchar（50） | 组件表名 |
| material\_id | bigint | 素材id |
| material\_type | varchar（50） | 素材表名 |
| create\_time | timestamp | 创建时间 |
| update\_time | timestamp | 最近更新时间 |

### 学生考试记录表

学生考试记录表mysql数据库设计student\_paper\_record

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 说明 |
| id | bigint | 主键 |
| student\_id | bigint | 学生id |
| source\_paper\_id | bigint | 原试卷id |
| mongo\_id | varchar | mongo做题记录试卷id |
| start\_time | timestamp | 开始时间 |
| submit\_time | timestamp | 提交时间 |
| create\_time | timestamp | 创建时间 |
| update\_time | timestamp | 最近更新时间 |

学生试卷记录mongodb集合：student\_papeer\_record

|  |
| --- |
| {  "\_id" : objectid("5b32f9b40c37c5b307ec7967"),  "structure" : [  {  "order" : "题号",  "type" : "类型：（选择题，填空题，判断题等）",  “answers”:[  {  "order" : "题号",  "question\_id" : "试题id",  "score" : "分数"，  “answer”:”标准答案”,  “student\_answer”:”学生答案”  },  {  "order" : "题号",  "question\_id" : "试题id",  "score" : "分数"，  “answer”:”学生答案”  }  ]  },  ],  "answers" : [  {  "question\_id" : "试题id",  "answer" : "答案a"  },  {  "question\_id" : "试题id",  "answer" : "答案b"  }  ]  } |

### 学生答题情况表（用于数据分析）：student\_answer

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 说明 |
| id | bigint | 主键 |
| question\_id | bigint | 题目id |
| question\_type | tinyint | 题目类型 |
| question\_order | int | 题号（需要约束） |
| answer | varchar(100) | 标准答案 |
| student\_answer | varchar(100) | 学生答案 |
| record\_id | bigint | 考试记录id |
| is\_true | tinyint | 是否正确 |
| postil | varchar(300) | 批注 |
| create\_time | timestamp | 创建时间 |
| update\_time | timestamp | 最近更新时间 |

### 组件文件夹：module\_folder

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 说明 |
| id | bigint | 主键 |
| name | varchar(20) | 文件夹名 |
| parent\_id | bigint | 上级目录id（根为空） |
| create\_time | timestamp | 创建时间 |
| create\_user | bigint | 创建人 |
| update\_user | bigint | 最后修改人 |
| deleted | smallint | 是否已经删除 |
| create\_time | timestamp | 创建时间 |
| update\_time | timestamp | 最近更新时间 |

# 功能模块划分

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 系统 | 模块 | 子模块 | 功能 | 功能点 |
| 题库系统 | 知识体系 | 主知识体系管理 | 添加主知识体系 | 暂不划分（下同） |
|  | 添加主知识点 |  |
|  | 修改主知识点名称 |  |
|  | 移动主知识点 |  |
|  | 删除主知识点 |  |
|  | 知识体系管理 | 添加知识体系 |  |
|  | 添加知识点 |  |
|  | 删除知识点 |  |
|  | 知识体系搜索 | 知识体系列表 |  |
|  | 知识体系详情 |  |
|  | 知识点详情 |  |
|  | 知识体系树详情 |  |
|  | 素材 | 题目管理 | 添加题目 |  |
|  | 编辑题目 |  |
|  | 添加衍生题 |  |
|  | 题目审核 |  |
|  | 题目列表 |  |
|  | 题目详情 |  |
|  | 视频管理 | 添加视频 |  |
|  | 编辑视频 |  |
|  | 视频列表 |  |
|  | 视频详情 |  |
|  | 知识素材管理 | 添加知识素材 |  |
|  | 编辑知识素材 |  |
|  | 知识素材列表 |  |
|  | 知识素材详情 |  |
|  | 组件 | 组件文件夹管理 | 添加组件文件夹 |  |
|  | 编辑组件文件夹 |  |
|  | 删除组件文件夹 |  |
|  | 讲义管理 | 添加讲义 |  |
|  | 编辑讲义 |  |
|  | 删除讲义 |  |
|  | 移动组件到文件夹 |  |
|  | 查看讲义列表 |  |
|  | 查看讲义详情 |  |
|  | 预览讲义 |  |
|  | 试卷管理 | 添加试卷 |  |
|  | 编辑试卷 |  |
|  | 删除试卷 |  |
|  | 移动组件到文件夹 |  |
|  | 查看试卷列表 |  |
|  | 查看试卷详情 |  |
|  | 预览试卷 |  |
|  | 课程 | 课程部分需求不明确 |  |  |
|  | 标签 | 标签部分的前端功能本期暂不实现 |  |  |

# 数据结构

## 标签数据结构设计

### 标签数据结构

以来源为例：

source\_item表数据：

|  |  |  |  |
| --- | --- | --- | --- |
| id | 111 | source\_id | 123 |
| tag\_name1 | name1 | tag\_value1 | value1 |
| tag\_name2 | name2 | tag\_value2 | value2 |
| tag\_name3 | name3 | tag\_value3 | value3 |
| tag\_name4 | name4 | tag\_value4 | value4 |

Json数据：

{“id”:111,”type”:”source”,“source\_id”:123,”name1”:”value1”,”name2:”value2”,”name3”:”value3”,”name4”:”value4”}

### 文本标签数据结构

一个对象可以有多个文本标签，对象和标签的对应关系存储在mongodb中，使用JsonArray存储，举例如下：

{“text\_label”:[“标签1”,”标签2”,”标签3”]}

## 素材数据结构设计

### 编辑器数据结构

Json结构，由前端编辑器产生，后端直接存储，不进行加工。

### 渲染引擎数据结构

Json结构，由编辑器数据结构转化而来。第一级包含line和object两个属性，line是一个jsonarray，里面的对象表示一个逻辑行，逻辑行也是一个jsonarray，里面的对象表示文本或图片等信息，包含类型type和索引key。Object也是一个jsonarray，里面的对象表示line中对象的信息，根据对象的key值进行关联。实例如下：

{

“Line”:

[

[ //一个逻辑行

{“type”:”text”,”key”:”text\_001”}, //一个文本对象

{“type”:”image”,”key”:”image\_001”} //一个图片对象

],

[…],

[…]

]

“Object”:

[

”text\_001”:{”content”:”abcd”}, //key值为text\_001的对象信息

”image\_001”:{”url”:”www.abc.cn”,”width”:”100px”,”height”:”200px”},

…

]

}

### 题目数据结构

这里主要说明mongodb中题干、提问、答案、解析的数据结构，这四部分都由编辑器数据结构构成，题干是一个编辑器数据结构，提问、答案和解析都是一个编辑器数据结构数组。这里说明一下提问和答案的设计原因。

提问表示选择题选项，填空题空白和问答题的问题，在一个题目中，选项、空白或问题都有可能是多个，因此采用数组。每一个选项、空白或问题都由编辑器提供，因此使用编辑器数据结构。

答案可能有多个，因此采用数组，每个答案都由编辑器提供，因此使用编辑器数据结构（为了统一，选择题也采用该结构）。

示例如下：

题干：

“content”:{一个编辑器数据结构}

提问：

“question”:[

{一个编辑器数据结构},{一个编辑器数据结构},{一个编辑器数据结构}

]

答案：

“result”:[

{一个编辑器数据结构},{一个编辑器数据结构},{一个编辑器数据结构}

]

解析：

“analysis”:[

{一个编辑器数据结构},{一个编辑器数据结构},{一个编辑器数据结构}

]

### 知识素材数据结构

知识素材的正文由编辑器提供，采用编辑器数据结构。

# 功能设计

## 标签功能设计

### 给文档打标签（内部API）

输入：文档类型，文档id，标签信息

说明：要建立标签信息类，包含类型，值和权重信息。

流程：

1、检查输入，是否合法。

2、根据标签类型查询标签分类。

3、按标签分类更新文档和标签值对应表。

输出：新增文档和标签值对应表记录主键，使用抛异常方式报错误。

定义：public Long addDocTag(String docType,Long docId,TagInfo tag) throw Exception;

标签信息TagInfo类定义：

Public class TagInfo{

Private Long type; //标签类型的id

Private Object item; //标签值（结构化标签为id主键，非结构化标签为文本）

Private Integer weight; //标签权重

}

### 给文档打多个标签（内部API）

输入：文档类型，文档id，包含标签信息的列表

流程：

1、检查输入，是否合法。

2、根据标签类型查询标签分类。

3、将标签按标签分类分组整理。

4、按标签分类批量更新文档和标签值对应表。

输出：void类型，使用抛异常方式报错误。

定义：public void addDocTagBatch(String docType,Long docId,List<TagInfo> tagList) throw Exception;

### 根据标签搜索文档（内部API）

输入：文档类型，搜索的值域列表，包含标签信息的列表

流程：

1、检查输入，是否合法。

2、值域和标签数组初始化es搜索条件。

3、在es中搜索，并返回结果。

输出：文档列表，使用抛异常方式报错误。

定义：public List searchDoc(String doctype,List<TagInfo> tagList) throw Exception;

### 添加一层标签类型（内部API）

输入：标签名称，标签分类，搜索key值

说明：只开放一层标签的自定义，其余类型不开放。

流程：

1、检查输入，是否合法。

2、在标签类型表中添加数据。

输出：新增标签类型记录的主键，使用抛异常方式报错误。

定义：public Long addTagType(String name,Integer classify,String searchKey) throw Exception;

### 修改一层标签类型（内部API）

输入：标签类型id，标签名称，搜索key值

说明：只开放一层标签的修改且只能修改自定义标签，其余类型不开放。

流程：

1、检查输入，是否合法。

2、在标签类型表中修改数据。

输出：void类型，使用抛异常方式报错误。

定义：public void changeTagType(Long tagTypeId,String name,String searchKey) throw Exception;

### 添加一层标签值（内部API）

输入：标签类型id，标签值

流程：

1、检查输入，是否合法。

2、在标签值表中添加数据。

输出：新增标签值记录的主键，使用抛异常方式报错误。

定义：public Long addTagItem(String name,Long tagTypeId) throw Exception;

### 添加来源标签（内部API）

输入：文档类型，标签值列表

流程：

1、检查输入，是否合法。

2、在来源标签表中添加数据。

输出：新增来源标签的主键列表，使用抛异常方式报错误。

定义：public List<Long> addSourceTag(String docType,List<SourceItem> tagList) throw Exception;

SourceItem类是来源标签表的映射类。

### 添加文本标签值（内部API）

输入：文档类型，标签值列表，用户id

流程：

1、检查输入，是否合法。

2、在文本标签表中添加数据。

输出：void类型，使用抛异常方式报错误。

定义：public void addTextTag(String docType,List<String> nameList,Long userId) throw Exception;

## 知识体系功能设计

### 添加主知识体系

输入：学科id和学部id

流程：

1、检查输入，是否合法，学部和学科是否对应。

2、检查是否已存在学部和学科相同的主知识体系。

3、主知识体系表插入数据。

输出：是否成功的提示

前端接口：addMainHierarchy

参数：{“subject”:1,”department”:1}

返回值：{“code”:1}

接口统一使用code传递结果，后端维护结果编码表，前端根据编码表查询结果的含义。

### 添加主知识点

输入：主知识体系id，上层知识点id（第一层为空），知识点名称

流程：

1、检查输入，是否合法。

2、检查是否存在同名知识点。

3、主知识点表插入数据。

4、修改对应主知识体系的更新时间。

5、知识点表插入数据。

6、修改对应知识体系的更新时间。

输出：是否成功的提示

前端接口：addMainPoint

参数：{“hierarchyId”1,“parentId”:1,”name”:”主知识点”}

返回值：{“code”:1}

### 修改主知识点名称

输入：主知识体系id，主知识点id，修改的名称

流程：

1、检查输入，是否合法。

2、检查是否存在同名知识点。

3、修改对应主知识点数据。

4、修改对应主知识体系的更新时间。

5、修改对应知识点数据。

6、修改对应知识体系的更新时间。

输出：是否成功的提示

前端接口：editMainPointName

参数：{“hierarchyId”1,“pointId”:1,”name”:”主知识点1”}

返回值：{“code”:1}

### 移动主知识点

输入：主知识体系id，主知识点id，移动的父节点id

流程：

1、检查输入，是否合法。

2、检查是否存在同名知识点。

3、修改对应主知识点数据。

4、修改主知识点的子节点数据。

5、修改对应主知识体系的更新时间。

3、修改对应知识点数据。

4、修改知识点的子节点数据。

5、修改对应知识体系的更新时间。

输出：是否成功的提示

前端接口：moveMainPoint

参数：{“hierarchyId”1,“pointId”:2,”parentId”:1}

返回值：{“code”:1}

### 添加知识体系

输入：学科id、学部id、知识体系名称

流程：

1、检查输入，是否合法，学部和学科是否对应。

2、检查是否已存在学部、学科和名称都相同的知识体系。

3、知识体系表插入数据。

输出：是否成功的提示

前端接口：addMainHierarchy

参数：{“subject”:1,”department”:1,”name”:”知识体系”}

返回值：{“code”:1}

### 添加知识点

输入：知识体系id，主知识点id

流程：

1、检查输入，是否合法。

2、知识点修改为使用。

3、修改对应知识体系的更新时间。

输出：是否成功的提示

前端接口：addPoint

参数：{“hierarchyId”1,“mainPointId”:1}

返回值：{“code”:1}

### 删除知识点

输入：知识体系id，知识点id

流程：

1、检查输入，是否合法。

2、知识点修改为不使用。

3、知识点的子节点改为不使用

4、修改对应知识体系的更新时间。

输出：是否成功的提示

前端接口：removePoint

参数：{“hierarchyId”1,“pointId”:1}

返回值：{“code”:1}

### 知识体系列表

说明：将mysql中知识体系和主知识体系数据同步到es，在es中共用一个index，使用type进行区分，搜索在es中进行。

输入：要搜索的学科id和学部id

流程：

1、检查输入，是否合法。

2、按学科和学部搜索。

输出：知识体系列表

前端接口：listHierarchy

参数：{“subject”:1,”department”:1}

返回值：{“code”:1,data:[{知识体系列表内容},{知识体系列表内容}]}

### 知识体系详情

输入：知识体系id，知识体系类型

流程：

1、检查输入，是否合法。

2、取出对应知识体系的第一层知识点。

输出：第一层知识点列表

前端接口：detailHierarchy

参数：{“hierarchyId”:1,”type”:1} （0表示主知识体系，1表示知识体系）

返回值：{“code”:1,data:{知识体系详情内容}}

### 知识点详情

输入：知识体系id，知识体系类型，知识点id

流程：

1、检查输入，是否合法。

2、取出对应知识点的下层知识点。

输出：下层知识点列表

前端接口：detailPoint

参数：{“hierarchyId”:1,”pointId”:1,”type”:1} （0表示主知识体系，1表示知识体系）

返回值：{“code”:1,data:[{下层知识点内容},{下层知识点内容},{下层知识点内容}]}

### 知识体系树详情

输入：知识体系id，知识体系类型

流程：

1、检查输入，是否合法。

2、取出对应知识体系的知识点树。

输出：知识点树（封装json）

前端接口：detailHierarchyTree

参数：{“hierarchyId”:1,”type”:1} （0表示主知识体系，1表示知识体系）

返回值：{“code”:1,data:[{知识点内容},{知识点内容},{知识点内容}]}

说明：前端可按照深度和父节点id遍历树。

## 素材功能设计

### 添加题目

输入：学科id、学部id、范围、题型、年份、难度、层次、来源标签、知识点标签、文本标签、题干、提问、答案、解析

流程：

1、检查输入，是否合法。

2、建立来源标签，写入mysql来源标签表。

3、建立文本标签，写入mysql文本标签表。

4、建立题目数据，包含题型，写入mysql题目表。

5、建立题目内容数据，包含mysql id，题干、提问、答案、解析、文本标签，写入mongodb题目集合。

6、更新mysql题目表数据，写入mongodb id。

7、建立题目和标签的关联，在文档和标签值对应表中添加数据。

输出：是否成功的提示

前端接口：addQuestion

参数：{“subjectId”:1,”departmentId”:1,”scopeId”:1,”type”:0,”year”:1,”degree”:1,”level”:1,”source”:[{“id”:1,”type1”:”页”,”item1”:”号”}],”point”:[{“id”:1,”type”:0,”weight”:10}],”text”:[“a”,”b”,”c”], “content”:{一个编辑器数据结构},“question”:[{一个编辑器数据结构},{一个编辑器数据结构},{一个编辑器数据结构}],“result”:[{一个编辑器数据结构},{一个编辑器数据结构},{一个编辑器数据结构}],“analysis”:[{一个编辑器数据结构},{一个编辑器数据结构},{一个编辑器数据结构}]}

返回值：{“code”:1}

### 编辑题目

输入：题目id，学科id、学部id、范围、题型、年份、难度、层次、来源标签、知识点标签、文本标签、题干、提问、答案、解析

流程：

1、检查输入，是否合法。

2、建立来源标签，写入mysql来源标签表。

3、建立文本标签，写入mysql文本标签表。

4、建立题目内容数据，包含mysql id，题干、提问、答案、解析、文本标签，写入mongodb题目集合。

5、建立题目数据，包含题型、mongodb id，更新mysql题目表。

6、建立题目和标签的关联，在文档和标签值对应表操作，先将题目的原关联删除，再添加数据。

输出：是否成功的提示

前端接口：editQuestion

参数：{“questionId”:1,“subjectId”:1,”departmentId”:1,”scopeId”:1,”type”:0,”year”:1,”degree”:1,”level”:1,”source”:[{“id”:1,”type1”:”页”,”item1”:”号”}],”point”:[{“id”:1,”type”:0,”weight”:10}],”text”:[“a”,”b”,”c”], “content”:{一个编辑器数据结构},“question”:[{一个编辑器数据结构},{一个编辑器数据结构},{一个编辑器数据结构}],“result”:[{一个编辑器数据结构},{一个编辑器数据结构},{一个编辑器数据结构}],“analysis”:[{一个编辑器数据结构},{一个编辑器数据结构},{一个编辑器数据结构}]}

返回值：{“code”:1}

### 添加衍生题

输入：学科id、学部id、范围、题型、年份、难度、层次、来源标签、知识点标签、文本标签、题干、提问、答案、解析，母题id

流程：

1、检查输入，是否合法。

2、建立来源标签，写入mysql来源标签表。

3、建立文本标签，写入mysql文本标签表。

4、建立题目数据，包含题型、母题id，写入mysql题目表。

5、建立题目内容数据，包含mysql id，题干、提问、答案、解析、文本标签，写入mongodb题目集合。

6、更新mysql题目表数据，写入mongodb id。

7、建立题目和标签的关联，在文档和标签值对应表中添加数据。

输出：是否成功的提示

前端接口：addSubQuestion

参数：{“parentId”:1,“subjectId”:1,”departmentId”:1,”scopeId”:1,”type”:0,”year”:1,”degree”:1,”level”:1,”source”:[{“id”:1,”type1”:”页”,”item1”:”号”}],”point”:[{“id”:1,”type”:0,”weight”:10}],”text”:[“a”,”b”,”c”], “content”:{一个编辑器数据结构},“question”:[{一个编辑器数据结构},{一个编辑器数据结构},{一个编辑器数据结构}],“result”:[{一个编辑器数据结构},{一个编辑器数据结构},{一个编辑器数据结构}],“analysis”:[{一个编辑器数据结构},{一个编辑器数据结构},{一个编辑器数据结构}]}

返回值：{“code”:1}

### 题目列表

说明：题目信息会从mysql和mongodb同步到es，一个题目在es中使用两个索引，mongodb使用一个索引，包含正文和文本标签，主要用于关键词文本搜索，mysql使用一个索引，用于id的精确匹配。

输入：学科id、学部id、范围、题型、年份、难度、层次、来源、知识点id、关键字等搜索信息。

流程：

1、检查输入，是否合法。

2、搜索es中的mysql索引，找出对应的题目id。

3、搜索es中的mongodb索引，找出最终题目id。

4、按更新时间排序，在mysql中取出对应分页的数据。

输出：题目列表

前端接口：listQuestion

参数：{“subjectId”:1,”departmentId”:1,”scopeId”:1,”type”:0,”year”:1,”degree”:1,”level”:1,”source”:[{“id”:1,”type1”:”页”,”item1”:”号”}],”point”:[{“id”:1,”type”:0,”weight”:10}],”keyword”:”abc”}

返回值：{“code”:1,data:[{题目列表内容},{题目列表内容}]}

### 题目详情

输入：题目id。

流程：

1、检查输入，是否合法。

2、取出mysql中的题目数据。

3、取出mongodb中的题目数据。

输出：题目详情

前端接口：detailQuestion

参数：{“questionId”:1}

返回值：{“code”:1,data:{题目详情内容}}

### 添加视频

输入：学科id、学部id、题型、年份、知识点标签、文本标签、视频名称、主讲人、视频url

流程：

1、检查输入，是否合法。

2、建立文本标签，写入mysql文本标签表。

3、建立视频数据，包含视频名称、主讲人、视频url，写入mysql视频表。

4、建立视频内容数据，包含mysql id，视频信息、文本标签，写入mongodb视频集合。

5、更新mysql视频表数据，写入mongodb id。

6、建立视频和标签的关联，在文档和标签值对应表中添加数据。

输出：是否成功的提示

前端接口：addVideo

参数：{“subjectId”:1,”departmentId”:1,”type”:0,”year”:1,”point”:[{“id”:1,”type”:0,”weight”:10}],”text”:[“a”,”b”,”c”],”name”:”test”,”speaker”:”abc”,”url”:”url”}

返回值：{“code”:1}

### 编辑视频

输入：视频id，学科id、学部id、题型、年份、知识点标签、文本标签、视频名称、主讲人、视频url

流程：

1、检查输入，是否合法。

2、建立文本标签，写入mysql文本标签表。

3、建立视频内容数据，包含mysql id，视频信息，写入mongodb视频集合。

4、建立视频数据，包含视频名称、主讲人、视频url、mongodb id，更新mysql视频表。

5、建立视频和标签的关联，在文档和标签值对应表中操作，先将视频的原关联删除，再添加数据。

输出：是否成功的提示

前端接口：editVideo

参数：{“videoId”:1,“subjectId”:1,”departmentId”:1,”type”:0,”year”:1,”point”:[{“id”:1,”type”:0,”weight”:10}],”text”:[“a”,”b”,”c”],”name”:”test”,”speaker”:”abc”,”url”:”url”}

返回值：{“code”:1}

### 视频列表

说明：视频信息会从mysql和mongodb同步到es，一个视频在es中使用两个索引，mongodb使用一个索引，包含正文和文本标签，主要用于关键词文本搜索，mysql使用一个索引，用于id的精确匹配。

输入：学科id、学部id、题型、年份、知识点标签、搜索关键词。

流程：

1、检查输入，是否合法。

2、搜索es中的mysql索引，找出对应的视频id。

3、搜索es中的mongodb索引，找出最终视频id。

4、按更新时间排序，在mysql中取出对应分页的数据。

输出：视频列表

前端接口：listVideo

参数：{“subjectId”:1,”departmentId”:1,”type”:0,”year”:1,”point”:[{“id”:1,”type”:0,”weight”:10}],”keyword”:”abc”}

返回值：{“code”:1,data:[{视频列表内容},{视频列表内容}]}

### 视频详情

输入：视频id。

流程：

1、检查输入，是否合法。

2、取出mysql中的视频数据。

3、取出mongodb中的视频数据。

输出：视频详情

前端接口：listVideo

参数：{“videoId”:1}

返回值：{“code”:1,data:{视频详情内容}}

### 添加知识素材

输入：学科id、学部id、年份、知识点标签、文本标签、名称、正文

流程：

1、检查输入，是否合法。

2、建立文本标签，写入mysql文本标签表。

3、建立知识素材数据，包含名称，写入mysql知识素材表。

4、建立知识素材内容数据，包含mysql id、正文、文本标签，写入mongodb视频集合。

5、更新mysql知识素材表数据，写入mongodb id。

6、建立知识素材和标签的关联，在文档和标签值对应表中添加数据。

输出：是否成功的提示

前端接口：addKnowledge

参数：{“subjectId”:1,”departmentId”:1,”year”:1,”point”:[{“id”:1,”type”:0,”weight”:10}],”text”:[“a”,”b”,”c”],”name”:”test”,” content”:{一个编辑器数据结构}}

返回值：{“code”:1}

### 编辑知识素材

输入：知识素材id，学科id、学部id、年份、知识点标签、文本标签、名称、正文

流程：

1、检查输入，是否合法。

2、建立文本标签，写入mysql文本标签表。

3、建立知识素材内容数据，包含mysql id、正文、文本标签，写入mongodb视频集合。

4、建立知识素材数据，包含名称、mongodb id，更新mysql视频表。

5、建立知识素材和标签的关联，在文档和标签值对应表中操作，先将知识素材的原关联删除，再添加数据。

输出：是否成功的提示

前端接口：editKnowledge

参数：{“knowledgeId”:1,“subjectId”:1,”departmentId”:1,”year”:1,”point”:[{“id”:1,”type”:0,”weight”:10}],”text”:[“a”,”b”,”c”],”name”:”test”,” content”:{一个编辑器数据结构}}

返回值：{“code”:1}

### 知识素材列表

说明：知识素材信息会从mysql和mongodb同步到es，一个知识素材在es中使用两个索引，mongodb使用一个索引，包含正文和文本标签，主要用于关键词文本搜索，mysql使用一个索引，用于id的精确匹配。

输入：学科id、学部id、年份、知识点标签、搜索关键词。

流程：

1、检查输入，是否合法。

2、搜索es中的mysql索引，找出对应的知识素材id。

3、搜索es中的mongodb索引，找出最终知识素材id。

4、按更新时间排序，在mysql中取出对应分页的数据。

输出：知识素材列表

前端接口：listKnowledge

参数：{“subjectId”:1,”departmentId”:1,”year”:1,”point”:[{“id”:1,”type”:0,”weight”:10}],”keyword”:”abc”}

返回值：{“code”:1,data:[{知识素材列表内容},{知识素材列表内容}]}

### 知识素材详情

输入：视频id。

流程：

1、检查输入，是否合法。

2、取出mysql中的知识素材数据。

3、取出mongodb中的知识素材数据。

输出：知识素材详情

前端接口：detailKnowledge

参数：{“knowledgeId”:1}

返回值：{“code”:1,data:{知识素材详情内容}}

## 组件功能设计

### 根据素材查询组件

输入：素材ID，素材类型

流程：

1. 判断素材ID，素材类型是否合法
2. 通过module\_material\_rel表根据素材ID，素材类型查询出所有组件

输出：查询到的组件

方法：public List<Module> findModuleByMaterial(Long materialId)

### 根据组件查询素材

输入：组件ID，组件类型

流程：

1. 判断组件ID是否合法
2. 通过module\_material\_rel表根据组件ID，组件类型查询出所有素材
3. 根据素材类型，在不同表中查询数据

输出：查询到的素材

方法：public List<Material> findMaterialByModule (Long moduleId)

### 添加素材和组件的关系

输入：组件ID，组件类型，素材ID，素材类型

流程：

1. 判断组件ID，组件类型，素材ID，素材类型是否合法
2. 添加module\_material\_rel表数据
3. 设置组件ID，组件类型，素材ID，素材类型
4. 保存数据

输出：是否保存成功

方法：public Boolean linkModuleAndMaterial(Long moduleId,String moduleType,Long materialId,String materialType)

### 添加组件和试题解析的关系

输入：组件ID，组件类型，试题解析ID

流程：

1. 判断组件ID，组件类型，试题解析ID合法性
2. 添加module\_analysis\_rel 表数据
3. 设置组件ID，组件类型，试题解析ID
4. 保存数据

输出：是否保存成功

方法：public Boolean linkModuleAndAnalysis (Long moduleId,String moduleType,Long materialId)

### 添加学生考试记录

输入：学生ID，试卷ID

流程：

1. 判断参数合法性
2. 添加student\_paper\_record表数据
3. 设置学生ID和试卷ID
4. 保存数据

输出：是否保存成功

方法：public Boolean addTestRecord(Long studentId,Long paperId)

### 根据考试记录查询做题记录

输入：考试记录ID

流程：

1. 判断参数合法性
2. student\_answer表中根据考试记录ID查询记录
3. 根据查询到的做题记录ID查询做题记录

输出：做题记录

方法：public List<AnswerRecord> findAnswerRecordByTest(Long testRecordId)

### 根据做题记录查询考试记录

输入：做题记录ID

流程：

1、判断参数合法性

1. 根据做题记录ID查询做题记录

3、根据做题记录获得考试记录ID

1. 根据考试记录ID查询考试记录表

输出：考试记录

方法：public List<TestRecord> findTestRecordByAnswer(Long answerRecordId)

### 从卷子复制出讲义

输入：卷子id，知识点讲解

流程：

1. 检查数据合法性
2. 将卷子内容复制到讲义
3. 讲义中添加知识点讲解（mysql和mongodb）
4. 保存讲义

输出：是否复制成功

前端接口：copyPaperToHandout

参数：{“paper”:1,”knowledgeExplain”:json}

返回值：{“code”:1}

### 保存学生做题

输入：卷子id，题目id，学生答案，学生试卷记录表id

流程：

条件：学生做卷子时，就保存学生试卷记录表id

1. 判断数据合法
2. 学生做题情况关联学生考试记录表
3. 保存学生做题情况表

输出：是否保存成功

前端接口：saveQuestionRecord

参数：{“paper”:1,”question”:1,”answer”:json,” studentPaperRecord”:1}

返回值：{“code”:1}

### 添加组件文件夹

输入：文件夹名称，上级文件夹id（可为空）

流程：

1. 判断文件夹名称是否合法。（包括重复命名判断）
2. 判断上级文件夹id是否存在且未被删除。
3. 保存文件夹

输出：插入结果

输出：是否保存成功

前端接口：addModuleFolder

参数：{“parentId”:1,”name”:文件夹名}

返回值：{“code”:1}

### 移动组件文件夹

输入：文件夹id，目标文件夹

流程：

1. 判断文件夹id是否存在
2. 判断目标文件夹是否存在
3. 修改mysql文件夹表中parent\_id字段
4. 保存数据

输出：移动结果

前端接口：moveModuleFolder

参数：{”folderId”:1, “parentId”:1}

返回值：{“code”:1}

### 编辑组件文件夹

输入：文件夹id，要修改的名称

流程：

1. 判断文件夹名称是否合法（包括重复命名）
2. 修改文件夹名
3. 保存数据

输出：编辑结果

前端接口：alterModuleFolder

参数：{”folderId”:1, “name”:文件夹名}

返回值：{“code”:1}

### 删除组件文件夹

输入：文件夹id

流程：

1. 判断文件夹中是否有组件（包括子文件夹中）
2. 提示是否连同组件一同删除
3. 修改mysql字段为删除状态

输出：删除结果

前端接口：deleteModuleFolder

参数：{”folderId”:1}

返回值：{“code”:1}

### 判断素材在该组件的文件夹中是否已经被使用

输入：素材id

流程：

1. 查询该组件所在文件夹中的所有组件
2. 查询所有组件中的所有素材
3. 判断素材是否已经存在

输出：返回素材是否被同文件夹中的组件使用

前端接口：isUsedInFolder

参数：{” materialId”:1}

返回值：{“code”:1}

### 添加试卷

输入：文件夹id，组件名，素材内容

流程：

1. 判断试卷名是否合法。
2. 判断素材内容是否合法。
3. 判断文件夹id是否合法。
4. 保存素材内容到mongodb。
5. 保存组件名，文件夹id，mongodb中的组件id等内容到mysql。

输出：返回保存结果

前端接口：addTestPaper

参数：{” folderId”:1,”name”:名称,”content”:json}

返回值：{“code”:1}

### 编辑试卷

输入：试卷id，素材内容

流程：

1. 判断组件id是否存在
2. 判断素材内容是否合法
3. 关联试卷main\_id
4. 保存素材内容到mongodb
5. 保存试卷到myasql

输出：是否保存成功

前端接口：updateTestPaper

参数：{” paperId”:1,”content”:json}

返回值：{“code”:1}

### 添加讲义

输入：文件夹id，组件名，素材内容

流程：

1. 判断组件名是否合法。
2. 判断素材内容是否合法。
3. 判断文件夹id是否合法。
4. 保存素材内容到mongodb。
5. 保存组件名，组件类型，文件夹id，mongodb中的素材内容id等内容到mysql。

输出：返回保存结果

前端接口：addHandoutPaper

参数：{” folderId”:1,”name”:名称,”content”:json}

返回值：{“code”:1}

### 编辑讲义

输入：讲义id，素材内容

流程：

1. 判断讲义id是否存在
2. 判断素材内容是否合法
3. 关联讲义main\_id
4. 保存素材内容到mongodb
5. 保存讲义到myasql

输出：是否保存成功

前端接口：updateHandoutPaper

参数：{” paperId”:1,”content”:json}

返回值：{“code”:1}

### 移动组件到文件夹

输入：文件夹id，组件id

流程：

1. 判断文件夹id是否存在
2. 判断组件名是否与文件夹中的命名重复
3. 修改mysql中组件的文件夹id

输出：移动是否成功

前端接口：moveModule

参数：{” module”:1,” targetFolder”:1}

返回值：{“code”:1}

### 查看组件列表

输入：查询条件

流程：

1. 从mysql中查询所有符合条件的组件

输出：查询结果

前端接口：findAllModule

参数：{” knowledge”:1,” 学科”:1,学部等标签}

返回值：List<Module>

### 学生考试

输入：试卷id，试题id，学生答案

流程：

1. 判断试卷id，试题id是否合法
2. 查询试卷答案
3. 对比学生答案判断对错
4. 组装该试题到学生试卷记录表
5. 关联学生考试记录
6. 添加数据到学生答题情况表

输出：客观题成绩

前端接口：findAllModule

参数：{” knowledge”:1,” 学科”:1,学部等标签}

返回值：List<Module>

### 数据统计

输入：组件id

流程：

1. 判断组件id是否合法
2. 查询学生做题情况表中该组件有关的数据
3. 组装统计数据（卷子平均难度需要冗余）

输出：统计的各种数据

前端接口：statistics

参数：{” paperId”:1}

返回值：json