ThinkGenius2 API 文档

概述

ThinkGenius2 是一个基于 Spring Boot 的智能块管理系统，提供块管理、AI 服务、关系管理和位置算法等功能。

基础信息

- **基础URL**: `http://localhost:8081`
- **API前缀**: `/api`
- **跨域支持**: 已配置，支持所有来源

认证

所有API都需要在请求参数中提供 `userId` 进行用户身份验证。

API 接口

1. 块管理 (Blocks)

1.1 创建块

```
POST /api/blocks?userId={userId}
Content-Type: application/json

{
  "type": "question|keyword|text",
  "position": {
    "x": 100.0,
    "y": 100.0
  },
  "size": {
    "width": 300.0,
    "height": 150.0
  },
  "content": "块内容"
}
```

1.2 创建块并自动计算位置

```
POST /api/blocks/auto-position?userId={userId}
Content-Type: application/json

{
  "type": "question|keyword|text",
  "size": {
    "width": 300.0,
    "height": 150.0
  },
  "content": "块内容"
}
```

1.3 获取所有块

```
GET /api/blocks?userId={userId}
```

1.4 根据类型获取块

```
GET /api/blocks/type/{type}?userId={userId}
```

1.5 根据ID获取块

```
GET /api/blocks/{id}?userId={userId}
```

1.6 更新块

```
PUT /api/blocks/{id}?userId={userId}
Content-Type: application/json

{
  "type": "question|keyword|text",
  "position": {
    "x": 100.0,
    "y": 100.0
  },
  "size": {
    "width": 300.0,
    "height": 150.0
  },
  "content": "更新后的内容"
}
```

1.7 更新块位置

```
PATCH /api/blocks/{id}/position?userId={userId}
Content-Type: application/json

{
  "x": 100.0,
  "y": 100.0
}
```

1.8 删除块

```
DELETE /api/blocks/{id}?userId={userId}
```

1.9 删除用户的所有块

```
DELETE /api/blocks?userId={userId}
```

1.10 重新布局所有块

```
POST /api/blocks/relayout?userId={userId}
```

1.11 检查位置是否可用

```
POST /api/blocks/check-position?userId={userId}
Content-Type: application/json

{
  "x": 100.0,
  "y": 100.0,
  "width": 200.0,
  "height": 100.0,
  "excludeBlockId": "optional-block-id"
}
```

1.12 找到最近的可用位置

```
POST /api/blocks/find-nearest-position?userId={userId}
Content-Type: application/json

{
  "x": 100.0,
  "y": 100.0,
  "width": 200.0,
  "height": 100.0,
  "excludeBlockId": "optional-block-id"
}
```

1.13 获取画布配置

```
GET /api/blocks/canvas-config
```

2. 块关系管理 (Relations)

2.1 创建关系

```
POST /api/relations?userId={userId}
Content-Type: application/json

{
  "sourceBlockId": "block1-id",
  "targetBlockId": "block2-id",
  "relationType": "related|parent|child|similar"
}
```

2.2 根据ID获取关系

```
GET /api/relations/{id}?userId={userId}
```

2.3 获取用户的所有关系

```
GET /api/relations?userId={userId}
```

2.4 获取与指定块相关的所有关系

```
GET /api/relations/block/{blockId}?userId={userId}
```

2.5 根据关系类型获取关系

```
GET /api/relations/type/{relationType}?userId={userId}
```

2.6 更新关系类型

```
PUT /api/relations/{id}?userId={userId}
Content-Type: application/json

{
  "relationType": "related|parent|child|similar"
}
```

2.7 删除关系

```
DELETE /api/relations/{id}?userId={userId}
```

2.8 删除与指定块相关的所有关系

```
DELETE /api/relations/block/{blockId}?userId={userId}
```

2.9 删除用户的所有关系

```
DELETE /api/relations?userId={userId}
```

2.10 检查两个块之间是否存在关系

```
POST /api/relations/exists?userId={userId}
Content-Type: application/json

{
  "sourceBlockId": "block1-id",
  "targetBlockId": "block2-id"
}
```

2.11 获取两个块之间的关系

```
POST /api/relations/between?userId={userId}
Content-Type: application/json

{
  "sourceBlockId": "block1-id",
  "targetBlockId": "block2-id"
}
```

2.12 批量创建关系

```
POST /api/relations/batch?userId={userId}
Content-Type: application/json

[
  {
    "sourceBlockId": "block1-id",
    "targetBlockId": "block2-id",
    "relationType": "related"
  },
  {
    "sourceBlockId": "block2-id",
    "targetBlockId": "block3-id",
    "relationType": "parent"
  }
]
```

2.13 获取关系统计信息

```
GET /api/relations/stats?userId={userId}
```

3. AI 服务 (AI)

3.1 生成关键词

```
POST /api/ai/keywords
Content-Type: application/json

{
  "content": "要分析的文本内容"
}
```

3.2 生成问题

```
POST /api/ai/questions
Content-Type: application/json

{
  "content": "要生成问题的文本内容"
}
```

3.3 生成摘要

```
POST /api/ai/summary
Content-Type: application/json

{
  "content": "要生成摘要的文本内容"
}
```

3.4 分析内容

```
POST /api/ai/analyze
Content-Type: application/json

{
  "content": "要分析的文本内容"
}
```

3.5 生成建议

```
POST /api/ai/suggestions
Content-Type: application/json

{
  "content": "要生成建议的文本内容"
}
```

3.6 聊天接口

```
POST /api/ai/chat
Content-Type: application/json

{
  "message": "用户消息"
}
```

4. 用户认证 (Auth)

4.1 用户注册

```
POST /api/auth/register
Content-Type: application/json

{
  "username": "用户名",
  "password": "密码",
  "email": "邮箱"
}
```

4.2 用户登录

```
POST /api/auth/login
Content-Type: application/json

{
  "username": "用户名",
  "password": "密码"
}
```

4.3 获取用户信息

```
GET /api/auth/user
Authorization: Bearer {jwt-token}
```

5. 测试接口 (Test)

5.1 健康检查

```
GET /api/test/ping
```

5.2 认证测试

```
GET /api/test/auth-test
```

5.3 块管理测试

```
GET /api/test/blocks-test
```

5.4 AI服务测试

```
GET /api/test/ai-test
```

数据模型

Block (块)

```
{
  "id": "块ID",
  "type": "question|keyword|text",
  "position": {
    "x": 100.0,
    "y": 100.0
  },
  "size": {
    "width": 300.0,
    "height": 150.0
  },
  "content": "块内容",
  "userId": "用户ID",
  "createdAt": "2024-01-01T00:00:00",
  "updatedAt": "2024-01-01T00:00:00"
}
```

BlockRelation (块关系)

```
{
  "id": "关系ID",
  "sourceBlockId": "源块ID",
  "targetBlockId": "目标块ID",
  "relationType": "related|parent|child|similar",
  "userId": "用户ID",
  "createdAt": "2024-01-01T00:00:00"
}
```

BlockRequest (块请求)

```
{
  "type": "question|keyword|text",
  "position": {
    "x": 100.0,
    "y": 100.0
  },
  "size": {
    "width": 300.0,
    "height": 150.0
  },
  "content": "块内容"
}
```

关系类型说明

- **related**: 相关关系，两个块内容相关
- **parent**: 父子关系，源块是目标块的父级
- **child**: 父子关系，源块是目标块的子级
- **similar**: 相似关系，两个块内容相似

块类型说明

- **question**: 问题块，用于提出问题
- **keyword**: 关键词块，用于标记重要概念
- **text**: 文本块，用于详细说明

错误处理

所有API都使用标准的HTTP状态码：

- `200 OK`: 请求成功
- `201 Created`: 创建成功
- `400 Bad Request`: 请求参数错误
- `401 Unauthorized`: 未授权

- `404 Not Found`: 资源不存在
- `500 Internal Server Error`: 服务器内部错误

配置说明

应用配置 (application.yml)

```yaml
server:
  port: 8081

spring:
  autoconfigure:
    exclude:
      - org.springframework.boot.autoconfigure.data.mongo.MongoDataAutoConfiguration
      - org.springframework.boot.autoconfigure.mongo.MongoAutoConfiguration
      - org.springframework.boot.autoconfigure.data.mongo.MongoRepositoriesAutoConfiguration

jwt:
  secret: your-jwt-secret
  expiration: 86400

baidu:
  qianfan:
    api-key: your-api-key
    secret-key: your-secret-key
    model: ERNIE-Bot-4
  api:
    access-token: your-access-token
```

测试

运行测试

```bash
# 编译项目
mvn clean compile

# 运行测试
mvn test

# 运行应用
mvn spring-boot:run
```

测试数据

项目包含多个测试类:

- `SimpleBlockPositionTest`: 位置算法测试
- `BlockRelationTest`: 关系管理测试

部署

构建JAR包

```
mvn clean package
```

运行JAR包

```
java -jar target/photo-ai-1.0-SNAPSHOT.jar
```

注意事项

1. 所有API都需要提供 `userId` 参数进行用户身份验证
2. 块删除时会自动删除相关的关系
3. 位置算法会自动避免块重叠
4. AI服务需要配置有效的百度文心一言API密钥
5. 建议在生产环境中配置MongoDB数据库