5.1 查询标签列表

5.1.1 需求

为了方便后期对文章进行管理，需要提供标签的功能，一个文章可以有多个标签。

在后台需要分页查询标签功能，要求能根据标签名进行分页查询。 **后期可能会增加备注查询等需求**。

注意：不能把删除了的标签查询出来。

5.1.2 标签表分析

通过需求去分析需要有哪些字段。

5.1.3 接口设计

请求方式	请求路径
Get	content/tag/list

Query格式请求参数：

```
pageNum: 页码
pageSize: 每页条数
name: 标签名
remark: 备注
```

响应格式：

```
{
    "code":200,
    "data":{
        "rows":[
            {
                "id":4,
                "name":"Java",
                "remark":"sdad"
            }
        ],
        "total":1
    },
    "msg":"操作成功"
}
```

5.1.4 代码实现

mybatis-plus-generator生成：tag表

Controller

```
@RestController
@RequestMapping("/content/tag")
public class TagController {
    @Autowired
    private TagService tagService;

    @GetMapping("/list")
    public ResponseResult<PageVo> list(Integer pageNum, Integer pageSize,
TagListDto tagListDto){
        return tagService.pageTagList(pageNum,pageSize,tagListDto);
    }
}
```

创建DTO：framework下domain/dto

```
@Data
@AllArgsConstructor
@NoArgsConstructor
public class TagListDto {
    private String name;
    private String remark;
}
```

Service

```
public interface ITagService extends IService<Tag> {

    ResponseResult<PageVo> pageTagList(Integer pageNum, Integer pageSize,
TagListDto tagListDto);
}
```

```
@Service("tagService")
public class TagServiceImpl extends ServiceImpl<TagMapper, Tag> implements
ITagService {

    @Override
    public ResponseResult<PageVo> pageTagList(Integer pageNum, Integer
pageSize, TagListDto tagListDto) {
        //分页查询
        LambdaQueryWrapper<Tag> queryWrapper = new LambdaQueryWrapper<>();

queryWrapper.eq(StringUtils.hasText(tagListDto.getName()),Tag::getName,tag
ListDto.getName());
```

```
queryWrapper.eq(StringUtils.hasText(tagListDto.getRemark()),Tag::getRemark
,tagListDto.getRemark());

        Page<Tag> page = new Page<>();
        page.setCurrent(pageNum);
        page.setSize(pageSize);
        page(page, queryWrapper);
        //封装数据返回
        PageVo pageVo = new PageVo(page.getRecords(),page.getTotal());
        return ResponseResult.okResult(pageVo);
    }
}
```

5.1.5 测试 - Postman

- pageNum、pageSize
- pageNum、pageSize、username, 检查idea console的数据库查询语句(取消yaml里的mybatis-plus日志功能注释)

5.2 新增标签

5.2.1 需求

点击标签管理的新增按钮可以实现新增标签的功能。

5.2.2 接口设计

请求方式	请求地址	请求头
POST	/content/tag	需要token请求头

请求体格式:

```
{"name":"c#","remark":"c++++"}
```

响应格式:

```
{
    "code":200,
    "msg":"操作成功"
}
```

5.2.3 代码实现

根据前面所学自主实现

5.2.4 测试

测试时注意，添加到数据库中的记录有没有 创建时间，更新时间，创建人，更新人字段。

5.3 删除标签

5.3.1 接口设计

请求方式	请求地址	请求头
DELETE	/content/tag/{id}	需要token请求头

请求参数在path中

例如：`content/tag/6`代表删除id为6的标签数据

响应格式：

```
{
    "code":200,
    "msg":"操作成功"
}
```

5.3.2 代码实现

根据前面所学自主实现

5.3.3 测试

注意测试删除后在列表中是否查看不到该条数据

数据库中该条数据还是存在的，只是修改了逻辑删除字段的值

5.4 修改标签

5.4.1 接口设计

5.4.1.1 获取标签信息

请求方式	请求地址	请求头
GET	/content/tag/{id}	需要token请求头

请求参数在path中

例如：content/tag/6 代表获取id为6的标签数据

响应格式：

```
{
    "code":200,
    "data":{
```

```
            "id":4,
            "name":"Java",
            "remark":"sdad"
        },
        "msg":"操作成功"
    }
```

5.4.1.2 修改标签接口

请求方式	请求地址	请求头
PUT	/content/tag	需要token请求头

请求体格式：

```
{"id":7,"name":"c#","remark":"c++++"}
```

响应格式：

```
{
    "code":200,
    "msg":"操作成功"
}
```

5.4.2 代码实现

根据前面所学自主实现

5.5 写博文

5.5.1 需求

需要提供写博文的功能，写博文时需要关联分类和标签。

可以上传缩略图，也可以在正文中添加图片。

文章可以直接发布，也可以保存到草稿箱。

5.5.2 表分析

标签和文章需要关联所以需要一张关联表：article_tag。

5.5.3 接口设计

思考下需要哪些接口才能实现这个功能?

5.5.3.1 查询所有分类接口

请求方式	请求地址	请求头
GET	/content/category/listAllCategory	需要token请求头

请求参数:

无

响应格式:

```
{
    "code":200,
    "data":[
        {
            "description":"wsd",
            "id":1,
            "name":"java"
        },
        {
            "description":"wsd",
            "id":2,
            "name":"PHP"
        }
    ],
    "msg":"操作成功"
}
```

5.5.3.2 查询所有标签接口

请求方式	请求地址	请求头
GET	/content/tag/listAllTag	需要token请求头

请求参数:

无

响应格式:

```
{
    "code":200,
    "data":[
        {
            "id":1,
            "name":"Mybatis"
        },
        {
            "id":4,
```

```
            "name":"Java"
        }
    ],
    "msg":"操作成功"
}
```

5.5.3.3 上传图片

请求方式	请求地址	请求头
POST	/upload	需要token请求头

参数：

img,值为要上传的文件

请求头：

Content-Type： multipart/form-data;

响应格式:

```
{
    "code": 200,
    "data": "文件访问链接",
    "msg": "操作成功"
}
```

5.5.3.4 新增博文

请求方式	请求地址	请求头
POST	/content/article	需要token请求头

请求体格式：

```
{
    "title":"测试新增博文",
    "thumbnail":"https://ptu-blog-oss.oss-cn-
beijing.aliyuncs.com/2022/08/21/4ceebc07e7484beba732f12b0d2c43a9.png",
    "isTop":"0",
    "isComment":"0",
    "content":"# 一级标题\n## 二级标题\n![Snipaste_20220228_224837.png]
(https://ptu-blog-oss.oss-cn-
beijing.aliyuncs.com/2022/08/21/c3af554d4a0f4935b4073533a4c26ee8.png)\n正
文",
    "tags":[
        1,
        4
```

```
    ],
    "categoryId":1,
    "summary":"哈哈",
    "status":"1"
}
```

响应格式：

```
{
    "code":200,
    "msg":"操作成功"
}
```

5.5.4 代码实现

5.5.4.1 查询所有分类接口

CategoryController

```
@RestController
@RequestMapping("/content/category")
public class CategoryController {
    @Autowired
    private CategoryService categoryService;

    @GetMapping("/listAllCategory")
    public ResponseResult listAllCategory(){
        List<CategoryVo> list = categoryService.listAllCategory();
        return ResponseResult.okResult(list);
    }
}
```

CategoryVo修改,增加description属性

```
@Data
@NoArgsConstructor
@AllArgsConstructor
public class CategoryVo {

    private Long id;
    private String name;
    //描述
    private String description;
}
```

CategoryService增加listAllCategory方法

```
public interface CategoryService extends IService<Category> {
    ResponseResult getCategoryList();
    List<CategoryVo> listAllCategory();
}
```

SystemConstants中增加常量

```
    /** 正常状态 */
    public static final String NORMAL = "0";
```

CategoryServiceImpl增加方法

```
    @Override
    public List<CategoryVo> listAllCategory() {
        LambdaQueryWrapper<Category> wrapper = new LambdaQueryWrapper<>();
        wrapper.eq(Category::getStatus, SystemConstants.NORMAL);
        List<Category> list = list(wrapper);
        List<CategoryVo> categoryVos = BeanCopyUtils.copyBeanList(list,
CategoryVo.class);
        return categoryVos;
    }
```

5.5.4.2 查询所有标签接口

TagVo

```
@Data
@AllArgsConstructor
@NoArgsConstructor
public class TagVo {
    private Long id;

    //标签名
    private String name;
}
```

TagController

```
    @GetMapping("/listAllTag")
    public ResponseResult listAllTag(){
        List<TagVo> list = tagService.listAllTag();
        return ResponseResult.okResult(list);
    }
```

TagService 增加listAllTag方法

```
List<TagVo> listAllTag();
```

TagServiceImpl

```
    @Override
    public List<TagVo> listAllTag() {
        LambdaQueryWrapper<Tag> wrapper = new LambdaQueryWrapper<>();
        wrapper.select(Tag::getId,Tag::getName);
        List<Tag> list = list(wrapper);
        List<TagVo> tagVos = BeanCopyUtils.copyBeanList(list,
TagVo.class);
        return tagVos;
    }
```

5.5.4.3 上传图片接口

在admin中增加UploadController

```
@RestController
public class UploadController {
    @Autowired
    private UploadService uploadService;

    @PostMapping("/upload")
    public ResponseResult uploadImg(@RequestParam("img") MultipartFile
multipartFile) {
        return uploadService.uploadImg(multipartFile);
    }
}
```

5.5.4.4 新增博文接口

ArticleController

```
@RestController
@RequestMapping("/content/article")
public class ArticleController {

    @Autowired
    private ArticleService articleService;
```

```
    @PostMapping
    public ResponseResult add(@RequestBody AddArticleDto article){
        return articleService.add(article);
    }
}
```

AddArticleDto

注意增加tags属性用于接收文章关联标签的id

```
@Data
@AllArgsConstructor
@NoArgsConstructor
public class AddArticleDto {

    private Long id;
    //标题
    private String title;
    //文章内容
    private String content;
    //文章摘要
    private String summary;
    //所属分类id
    private Long categoryId;

    //缩略图
    private String thumbnail;
    //是否置顶（0否，1是）
    private String isTop;
    //状态（0已发布，1草稿）
    private String status;
    //访问量
    private Long viewCount;
    //是否允许评论 1是，0否
    private String isComment;
    private List<Long> tags;
}
```

Article 修改这样创建时间创建人修改时间修改人可以自动填充

```
    @TableField(fill = FieldFill.INSERT)
    private Long createBy;
    @TableField(fill = FieldFill.INSERT)
    private Date createTime;
    @TableField(fill = FieldFill.INSERT_UPDATE)
    private Long updateBy;
    @TableField(fill = FieldFill.INSERT_UPDATE)
    private Date updateTime;
```

ArticleService增加方法

```
ResponseResult add(AddArticleDto article);
```

创建ArticleTag表相关的实体类，mapper，service,serviceimpl等

```
@TableName(value="article_tag")
@Data
@AllArgsConstructor
@NoArgsConstructor
public class ArticleTag implements Serializable {
    private static final long serialVersionUID = 625337492348897098L;

    /**
    * 文章id
    */
    private Long articleId;
    /**
    * 标签id
    */
    private Long tagId;
}
```

ArticleServiceImpl增加如下代码

```
    @Autowired
    private ArticleTagService articleTagService;

    @Override
    @Transactional
    public ResponseResult add(AddArticleDto articleDto) {
        //添加 博客
        Article article = BeanCopyUtils.copyBean(articleDto,
Article.class);
        save(article);

        List<ArticleTag> articleTags = articleDto.getTags().stream()
                .map(tagId -> new ArticleTag(article.getId(), tagId))
                .collect(Collectors.toList());

        //添加 博客和标签的关联
        articleTagService.saveBatch(articleTags);
        return ResponseResult.okResult();
    }
```