这是一个基于SSM（Spring + SpringMVC + MyBatis）框架的博客项目。以下是对项目结构的简要说明：

controller目录：

admin目录： 包含管理后台相关的控制器。

home目录： 包含前台（博客首页等）相关的控制器。

dto目录： 存放数据传输对象（DTO）。

entity目录： 存放实体类。

enums目录： 存放枚举类型。

interceptor目录： 存放拦截器。

mapper目录： 存放MyBatis的Mapper接口。

service目录： 存放服务接口及其实现类。

impl目录： 存放服务接口的实现类。

util目录： 存放工具类。

resources目录：

mapper目录： 存放MyBatis的XML映射文件。

mybatis目录： 存放MyBatis的配置文件。

spring目录： 存放Spring框架的配置文件。

webapp目录：

resource目录：

assets目录： 存放前端资源文件，如CSS、JS等。

WEB-INF目录：

view目录： 存放JSP视图文件，按照功能划分为Admin和Home两个部分。

test目录：

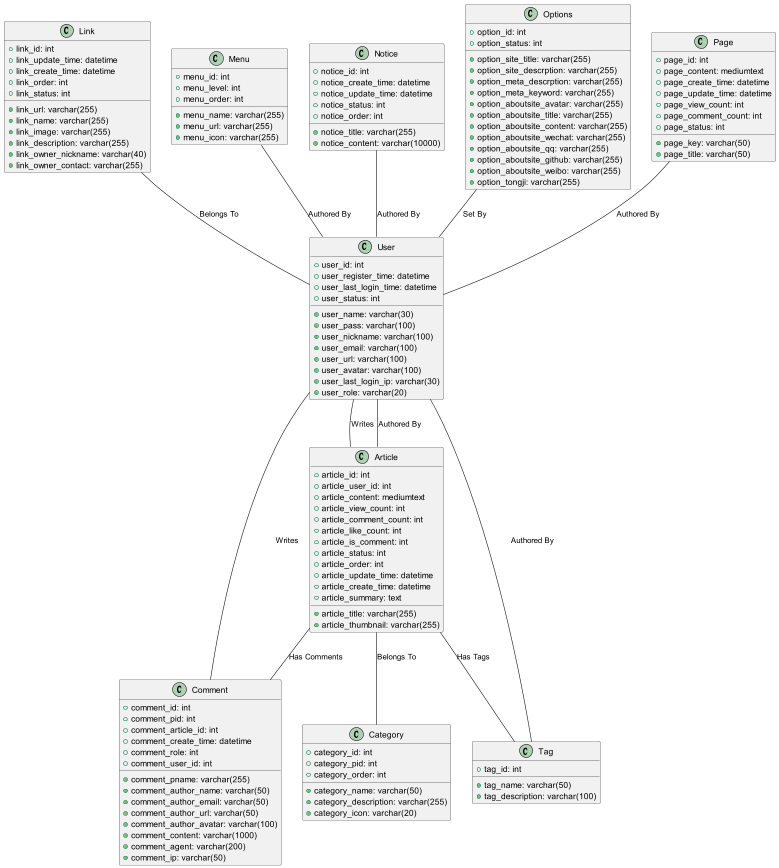
java目录：

com/Mingtc/ssm/blog目录：

mapper目录： 存放测试用的Mapper接口。

service目录： 存放测试用的服务接口。

项目的主要模块包括管理后台（Admin）和前台（Home），每个模块下又根据功能 划分了不同的包，包括文章、分类、评论、链接、菜单、公告、选项、页面、 标签、用户等。前端资源主要存放在webapp/resource/assets目录下，包括CSS、 JS等文件。 MyBatis 的映射文件存放在resources/mapper目录下。



用户实体（User Entity）：

Writes（写入）：表示用户与文章（Article表）和评论（Comment表）之间的关系。这意味着User表中的user\_id是Article表和Comment表的外键，将它们与创建它们的用户关联起来。

文章实体（Article Entity）：

Belongs To（属于）：表示Article表与Category表之间的关系。它表示一篇文章属于某个特定的分类。Article表中的category\_id是一个外键，引用了Category表的主键（category\_id）。

Has Tags（拥有标签）：表示Article表与Tag表之间的多对多关系。它暗示了存在一个名为article\_tag\_ref的连接表（在图中未显式显示），用于使用它们的相应ID将文章与标签关联起来。

Has Comments（拥有评论）：表示Article表与Comment表之间的关系。它表示一篇文章可以有多个相关的评论。Comment表中的comment\_article\_id是一个外键，引用了Article表中的article\_id。

Authored By（由...撰写）：表示文章由用户撰写。Article表中的article\_user\_id是一个外键，引用了User表中的user\_id。

链接实体（Link Entity）：

Belongs To（属于）：表示链接与用户之间的关系。Link表中的user\_id是一个外键，引用了User表中的user\_id。

菜单实体（Menu Entity）：

Authored By（由...撰写）：表示Menu表与User表之间的关系。它表示一个菜单由用户撰写。Menu表中的user\_id是一个外键，引用了User表中的user\_id。

公告实体（Notice Entity）：

Authored By（由...撰写）：表示一个公告由用户撰写。Notice表中的user\_id是一个外键，引用了User表中的user\_id。

选项实体（Options Entity）：

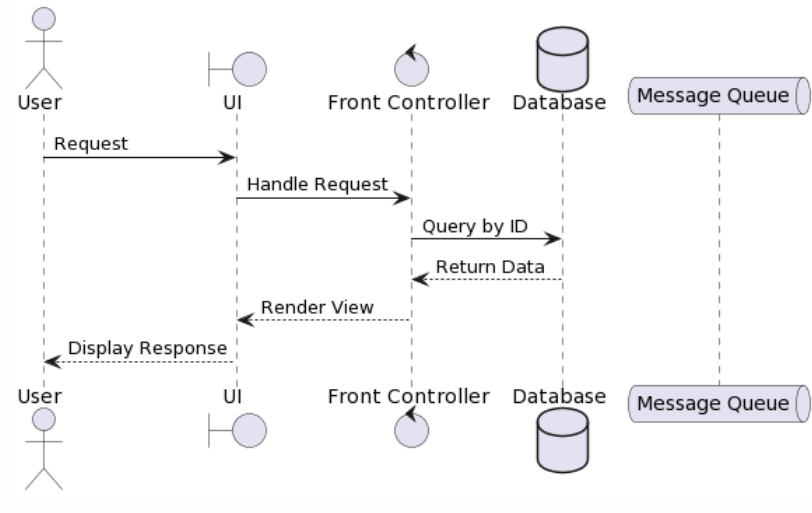
Set By（由...设置）：表示Options表与User表之间的关系。它表示站点选项由用户设置。Options表中的user\_id是一个外键，引用了User表中的user\_id。

页面实体（Page Entity）：

Authored By（由...撰写）：表示一个自定义页面由用户撰写。Page表中的user\_id是一个外键，引用了User表中的user\_id。

评论实体（Comment Entity）：

Authored By（由...撰写）：表示一个评论由用户撰写。Comment表中的comment\_user\_id是一个外键，引用了User表中的user\_id。



项目逻辑十分简单，运行顺序如下：

1. 用户 (User) 发送请求到用户界面 (UI)。
2. 用户界面 (UI) 将请求传递给前端控制器 (Front Controller)。
3. 前端控制器 (Front Controller) 发送查询请求给数据库 (Database)。
4. 数据库 (Database) 执行查询操作并返回数据给前端控制器 (Front Controller)。
5. 前端控制器 (Front Controller) 将数据渲染到用户界面 (UI)。
6. 用户界面 (UI) 显示响应给用户 (User)。