## 项目目录规范

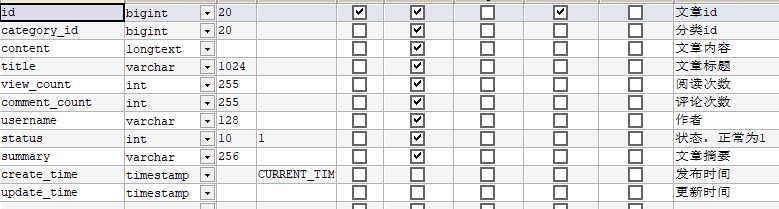


* model：实体
* dao：数据层
* service：业务逻辑
* controller：页面控制相关
* static：css js
* utils：工具
* views：HTML模板
* main.go：入口，定义路由

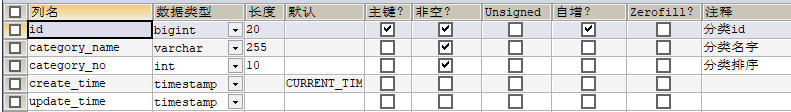
## 需求分析

## 数据库分析

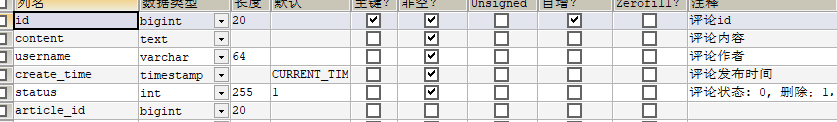
* article文章表



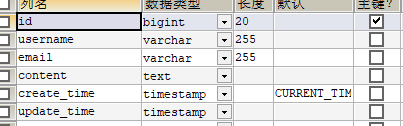
* category：分类表



* 评论表comment



* leave留言表



## 主页的实现

### 实体类

* 分类的结构体
* 文章的结构体（可以不写文章内容，因为是大文本）
* model/category.go

|  |
| --- |
| **package** model  *//`id``category\_name``category\_no``create\_time``update\_time`  // 定义分类结构体* **type** Category **struct** {  CategoryId int64 **`db:"id"`** CategoryName string **`db:"category\_name"`** CategoryNo int **`db:"category\_no"`** } |

* model/article.go

|  |
| --- |
| **package** model  **import "time"** *//``id``category\_id``content``title``view\_count``comment\_count``username``STATUS``summary``create\_time``update\_time``  // 定义文章结构体* **type** ArticleInfo **struct** {  Id int64 **`db:"id"`** CategoryId int64 **`db:"category\_id"`** *// 文章摘要* Summary string **`db:"summary"`** Title string **`db:"title"`** ViewCount uint32 **`db:"view\_count"`** *// 时间* CreateTime time.Time **`db:"create\_time"`** CommentCount uint32 **`db:"comment\_count"`** Username string **`db:"username"`** }  *// 用于文章详情页的实体 // 为了提升效率* **type** ArticleDetail **struct** {  ArticleInfo  *// 文章内容* Content string **`db:"content"`** Category }  *// 用于文章上下页* **type** ArticleRecord **struct** {  ArticleInfo  Category } |

### 数据层

* init()数据库初始化函数
* 分类相关的操作（添加，查询，查1个分类，查多个分类，查所有分类）
* 文章相关的操作（添加文章（投稿），查询所有文章，根据文章id查看内容）
* dao/db/db.go

|  |
| --- |
| **package** db  **import** (  \_ **"github.com/go-sql-driver/mysql"  "github.com/jmoiron/sqlx"** )  **var** (  DB \*sqlx.DB )  *// 初始化* **func** Init(dns string) error {  **var** err error  DB, err = sqlx.Open(**"mysql"**, dns)  **if** err != nil {  **return** err  }  *// 查看是否连成功* err = DB.Ping()  **if** err != nil {  **return** err  }  DB.SetMaxOpenConns(100)  DB.SetMaxIdleConns(16)  **return** nil } |

* dao/db/category.go

|  |
| --- |
| **package** db  **import "go\_5\_blog/model"** *// 添加分类* **func** InsertCategory(category \*model.Category) (categoryId int64, err error) {  sqlstr := **"insert into category(category\_name,category\_no) value(?,?)"** result, err := DB.Exec(sqlstr, category.CategoryName, category.CategoryNo)  **if** err != nil {  **return** }  categoryId, err = result.LastInsertId()  **return** }  *// 获取单个分类* **func** GetCategoryById(id int64) (category \*model.Category, err error) {  category = &model.Category{}  sqlstr := **"select id ,category\_name,category\_no from category where id = ?"** err = DB.Get(category, sqlstr, id)  **return** } |

### 业务逻辑层

### controller层

### 项目入口

## 投稿的实现

### 业务逻辑层

### controller层

## 文章详情页的实现

### 实体类

### 数据层

### 业务逻辑层

### controller层

## 文章评论的实现

### 实体类

### 数据层

### 业务逻辑层

### controller层

## 博客留言的实现

### 实体类

### 数据层

### 业务逻辑层

### controller层