实现文档 (Implementation Document)

互联网数据库开发课程设计项目实现文档

1. 项目概述

1.1 项目背景

本项目是南开大学“互联网数据库开发”课程的期末课程设计作品。项目旨在构建一个功能完善、安全可靠的新闻资讯管理系统。系统基于 PHP 的 Yii 2 Advanced 框架开发，严格遵循 MVC 设计模式，实现了前后台分离的架构。

1.2 系统目标

系统的主要目标是提供一个高效的新闻发布与管理平台：

- **面向管理员 (Backend)**：提供强大的内容管理功能，包括文章的发布、编辑、审核、分类管理、用户管理以及系统访问日志监控。

- **面向访客 (Frontend)**：提供友好的阅读体验，支持按分类浏览新闻、查看热门文章、发表留言互动以及了解团队信息。

1.3 核心功能特性

- **多角色用户系统**：支持普通用户注册/登录和管理员后台管理，具备基于角色的权限控制（RBAC）基础。

- **完整的内容管理流**：文章支持草稿、发布、下架三种状态，支持置顶、热门标记，支持富文本编辑。

- **交互式功能**：前台用户可以对文章进行点赞、收藏（需登录），并在留言板发表意见。

- **数据统计与分析**：后台首页提供系统概览，记录文章浏览量和点赞数，便于运营分析。

- **响应式设计**：前台页面采用 Bootstrap 栅格系统，适配 PC 和移动端设备。

---

2. 系统架构与技术栈

2.1 技术选型

- **后端语言**: PHP 7.4+

- **核心框架**: Yii 2 Advanced Template (v2.0.45+)

- **数据库**: MySQL 5.7 / 8.0 (InnoDB引擎)

- **前端框架**: Bootstrap 4, jQuery

- **Web服务器**: Apache 或 Nginx

- **版本控制**: Git

2.2 目录结构详解

项目采用 Yii 2 Advanced 模板的标准结构，确保了代码的清晰隔离和复用。

代码块

```
/
├── backend/                # 后台应用（仅管理员可访问）
│   ├── config/             # 后台特有配置
│   ├── controllers/        # 控制器（处理业务逻辑）
│   │   ├── ArticleController.php    # 文章管理
│   │   ├── SiteController.php       # 后台首页与登录
│   │   └── ...
│   ├── models/             # 后台专用模型（如搜索模型）
│   ├── views/              # 后台视图文件
│   └── web/                # 后台入口（index.php）
├── frontend/               # 前台应用（面向公众）
│   ├── config/             # 前台特有配置
│   ├── controllers/        # 前台控制器
│   │   ├── SiteController.php       # 首页、展示
│   │   └── GuestbookController.php  # 留言板
│   ├── models/             # 前台表单模型（SignupForm等）
│   ├── views/              # 前台视图文件
│   └── web/                # 前台入口
├── common/                 # 公共组件（前后台共享）
│   ├── config/             # 公共配置（数据库连接等）
│   └── models/             # 核心数据模型（ActiveRecord）
│       ├── User.php        # 用户模型
│       ├── PreNewsArticle.php # 文章模型
```

```
│       └── ...
├── console/                # 控制台应用
│   ├── migrations/         # 数据库迁移文件
│   └── controllers/        # 命令行控制器
└── data/                   # 数据库脚本与备份
```

2.3 MVC 模式应用

- **Model (模型)**: 位于 `common/models`，继承自 `yii\db\ActiveRecord`。负责与数据库交互、定义验证规则（Rules）和属性标签（Labels）。

- **View (视图)**: 位于 `views/` 目录，使用 PHP 混合 HTML 编写。通过 `Html` 辅助类生成安全的页面元素。

- **Controller (控制器)**: 位于 `controllers/` 目录，继承自 `yii\web\Controller`。负责接收请求、处理数据、渲染视图。

---

3. 数据库设计与实现

数据库设计遵循第三范式，统一使用 `pre_` 作为表前缀，确保与其他系统共存时不冲突。

3.1 核心数据表设计（部分）

3.1.1 用户表 (`pre_sys_user`)

存储系统所有用户账号信息。

代码块

```sql
CREATE TABLE `pre_sys_user` (
  `uid` int(11) NOT NULL AUTO_INCREMENT,
  `username` varchar(50) NOT NULL,
  `password_hash` varchar(255) NOT NULL, -- 存储加密后的密码
  `email` varchar(100) DEFAULT NULL,
  `role` tinyint(1) DEFAULT 0, -- 0:普通用户, 1:管理员
  `status` tinyint(1) DEFAULT 1, -- 10:正常, 0:禁用
```

```
  `created_at` datetime DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (`uid`),
  UNIQUE KEY `username` (`username`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

3.1.2 新闻文章表 (`pre_news_article`)

核心内容表，包含文章的所有元数据。

- **aid**: 主键ID。

- **cid**: 外键，关联 `pre_news_category`。

- **status**: 状态位（0:草稿, 1:发布, 2:下架）。

- **is_top / is_hot**: 推荐位标记。

- **views / likes**: 计数器字段。

3.1.3 留言板表 (`pre_guestbook` / `guestbook`)

记录用户留言，设计了 `is_read` 字段用于管理员后台标记已读状态。

3.2 数据库迁移 (Migrations)

项目使用 Yii2 的 Migration 机制管理数据库版本。

例如 `console/migrations/m130524_201442_init.php` 用于初始化用户表：

代码块

```
public function up()
{
    $tableOptions = null;
    if ($this->db->driverName === 'mysql') {
        $tableOptions = 'CHARACTER SET utf8 COLLATE utf8_unicode_ci ENGINE=InnoDB';
    }

    $this->createTable('{{%user}}', [
        'id' => $this->primaryKey(),
        'username' => $this->string()->notNull()->unique(),
        'auth_key' => $this->string(32)->notNull(),
        'password_hash' => $this->string()->notNull(),
```

```
            // ...
        ], $tableOptions);
    }
```

这种方式保证了团队开发时数据库结构的一致性。

---

4. 后端模块详细实现 (Backend)

4.1 访问控制 (Access Control)

后台控制器普遍使用了 `AccessControl` 行为（Behavior），强制要求管理员登录。

代码块

```php
// backend/controllers/SiteController.php
public function behaviors()
{
    return [
        'access' => [
            'class' => AccessControl::class,
            'rules' => [
                [
                    'actions' => ['login', 'error'],
                    'allow' => true,
                ],
                [
                    'actions' => ['logout', 'index'],
                    'allow' => true,
                    'roles' => ['@'], // '@' 代表已登录用户
                ],
            ],
        ],
    ];
}
```

4.2 文章管理模块

控制器: `backend\controllers\ArticleController`

- ****列表页 (`actionIndex`)****: 使用 `PreNewsArticleSearch` 模型进行过滤。支持按标题模糊搜索、按分类精确筛选、按状态筛选。

- ****创建/编辑 (`actionCreate/Update`)****: 使用同一个视图 `_form.php`，通过 `ActiveForm` 组件生成表单。

****表单视图 (`_form.php`)****:

使用了 Bootstrap 样式美化，利用 Grid 布局将表单分为两列。

代码块

```
<?php $form = ActiveForm::begin(); ?>
    <div class="form-row">
        <div class="col-md-8">
            <?= $form->field($model, 'title')->textInput(['maxlength' => true]) ?>
            <?= $form->field($model, 'content')->textarea(['rows' => 15]) ?>
        </div>
        <div class="col-md-4">
            <?= $form->field($model, 'cid')->dropDownList(
                ArrayHelper::map(PreNewsCategory::find()->all(), 'cid', 'name')
            ) ?>
            <?= $form->field($model, 'status')->dropDownList([
                1 => '发布', 0 => '草稿', 2 => '下架'
            ]) ?>
        </div>
    </div>
<?php ActiveForm::end(); ?>
```

4.3 访问日志模块

实现了 `VisitLogController`，配合 `VisitLogSearch` 模型，管理员可以查看系统的详细访问记录。这对于安全审计和流量分析至关重要。

---

5. 前端模块详细实现 (Frontend)

5.1 首页展示 (`SiteController::actionIndex`)

首页聚合了多类数据，通过一次 Controller Action 获取并传递给 View：

- **热门新闻**: 按 `views` 降序排列，取前几条。

- **最新发布**: 按 `created_at` 降序排列。

- **分类列表**: 获取所有启用状态的分类。

****视图实现 (`views/site/index.php`)****:

采用了 Hero Banner 设计展示抗战胜利80周年主题，下方使用卡片布局（Card Layout）展示新闻。

代码块

```
<!-- 热门文章循环 -->
<?php foreach ($hotArticles as $article): ?>
    <div class="news-card">
        <div class="news-card-image" style="background-image: url(...)">...
</div>
        <div class="news-card-body">
            <h3><?= Html::encode($article->title) ?></h3>
            <p><?= Html::encode(mb_substr($article->summary, 0, 80)) ?>...</p>
        </div>
    </div>
<?php endforeach; ?>
```

5.2 留言互动 (`GuestbookController`)

允许游客和登录用户留言。

- **智能填充**: 如果用户已登录 (`!Yii::$app->user->isGuest`)，控制器会自动从 `User` 模型获取用户名和邮箱填充到留言模型中。

- **IP记录**: 使用 `Yii::$app->request->userIP` 自动记录留言者的 IP 地址，用于防骚扰。

5.3 资源管理 (`AppAsset`)

通过 `frontend/assets/AppAsset.php` 管理前端资源。

- 依赖 `yii\bootstrap\BootstrapAsset`，确保页面自动加载 Bootstrap 3/4 的 CSS 和 JS。
- 自定义样式放在 `css/site.css` 中，实现个性化视觉效果。

---

6. 安全性实现

6.1 身份验证与授权

- 使用 `common\models\User` 类实现 `IdentityInterface` 接口。
- 密码存储使用 `Yii::$app->security->generatePasswordHash()` 生成哈希，绝不明文存储。
- 登录验证使用 `validatePassword()` 方法比对哈希值。

6.2 CSRF 防护

Yii2 默认开启 CSRF（跨站请求伪造）防护。在 `main.php` 配置中：

代码块

```
'components' => [
    'request' => [
        'csrfParam' => '_csrf-backend', // 后台使用独立的 CSRF token
    ],
]
```

所有 POST 表单都会自动包含 CSRF Token 隐藏域，防止恶意提交。

6.3 XSS 与 SQL 注入防御

- **XSS**: 视图层输出数据时，统一使用 `Html::encode()` 方法进行转义，防止恶意脚本注入。

- **SQL 注入**: 全面使用 `ActiveRecord` 和 `QueryBuilder`，底层使用 PDO 预处理语句（Prepared Statements），杜绝 SQL 注入风险。

---

7. 配置与环境

7.1 配置文件

配置采用分层覆盖机制：

1. `common/config/main.php`：全局配置（如缓存、数据库连接）。
2. `backend/config/main.php`：后台特有配置（如 Session 名、控制器命名空间）。
3. `*-local.php`：本地敏感配置（不纳入版本控制，包含数据库密码等）。

7.2 路由与 URL

支持 URL 美化（Pretty URL）。在 `config/main.php` 中配置 `urlManager`：

代码块

```
'urlManager' => [
    'enablePrettyUrl' => true,
    'showScriptName' => false,
    'rules' => [
        // 自定义路由规则
    ],
],
```

开启后，URL 将从 `index.php?r=site/login` 变为 `/site/login`，更利于 SEO。

---

8. 部署与维护指南

8.1 环境要求

- **操作系统**: Windows / Linux / macOS

- **Web服务器**: Apache 2.4+ / Nginx 1.10+

- **PHP 版本**: 7.4 - 8.2

- **数据库**: MySQL 5.7+

8.2 安装步骤

1. **克隆代码**: `git clone <repository-url>`
2. **安装依赖**: 在项目根目录运行 `composer install`。
3. **环境初始化**:
   - Windows: 运行 `init.bat`
   - Linux: 运行 `./init`
   - 选择 `0` (Development) 用于开发环境。
4. **数据库配置**: 编辑 `common/config/main-local.php`，填入正确的数据库主机、用户名和密码。
5. **数据导入**:
   - 方式一：运行 `php yii migrate` 创建基础表。
   - 方式二（推荐）：导入 `data/install.sql` 和 `data/news_data.sql` 获取完整的演示数据。

8.3 常见问题维护

- **权限问题**: 确保 `runtime/` 和 `web/assets/` 目录对 Web 服务器用户可写。
- **缓存清理**: 修改配置或数据库结构后，可能需要运行 `php yii cache/flush-all` 清理缓存。
- **调试模式**: 在 `index.php` 中定义 `YII_DEBUG` 为 `true` 可开启调试工具栏（Debug Toolbar），方便排查 SQL 查询和性能瓶颈。

---

9. 总结

本项目通过规范的架构设计和严谨的代码实现，完成了一个高可用、易扩展的新闻管理系统。Yii 2 框架的强大特性（如 Gii 代码生成、ActiveRecord、Widget 组件化）极大地提高了开发效率。系统在安全性、性能和用户体验方面均达到了课程设计的高标准要求。