学生学号	0121510870106	实验课成绩	

武汉理工大学

课 程 设 计 报 告 书

实验课程名称	XML 技术
开课学院	计算机科学与技术学院
指导教师姓名	冯静
学生姓名	谢泽丰
学生专业班级	软件 zy1502

2018 — 2019 学年 第 二 学期

实验课程名称：XML 技术

实验项目名称	网站新闻管理系统			实验成绩	
实验者	谢泽丰	专业班级	软件 zy1502	组别	
同组者				实验日期	2019 年 4 月 15 日

第一部分：课程设计内容描述

【设计题目】网站新闻管理系统

网站新闻管理系统又称为网站新闻信息发布系统，是将网页上的某些需要经常变动的信息，类似新闻、新产品发布和业界动态等更新信息集中管理，并通过信息的某些共性进行分类，最后系统化、标准化发布到网站上的一种网站应用程序。网站信息通过一个操作简单的界面加入数据库，然后通过已有的网页模板格式与审核流程发布到网站上。

【报告内容要求】

1、功能需求分析，了解需求。由于用例图是从用户角度来描述系统功能的，所以在进行需求分析时，使用用例图可以更好描述系统应具备什么功能。要求对新闻进行类别管理，内容管理、新闻搜索。

2、系统详细设计，包括前端显示页面，后台管理页面功能（增删修改新闻栏目、增删修改新闻内容、编辑新闻内容、用户管理;）。

3、系统数据实现，在实现本系统过程中，不可避免的要生成或存储一些持久性数据，如新闻类别、用户信息和新闻等。这些数据按传统方式一般是存储到数据库表中，而在系统我们要求将一部分比较小的数据或安全性低的数据存储到 XML 文件，如新闻信息;其他的数据继续存储在数据表中,数据库和 XML 文件可以进行相互转换。

4、系统功能实现并部署调试相关界面截图。

5、系统中涉及的 XML 文件及操作代码打印。

【思考】

1、如何优化设计好的系统，使其功能更完善？

2、设计一个系统要注意什么？

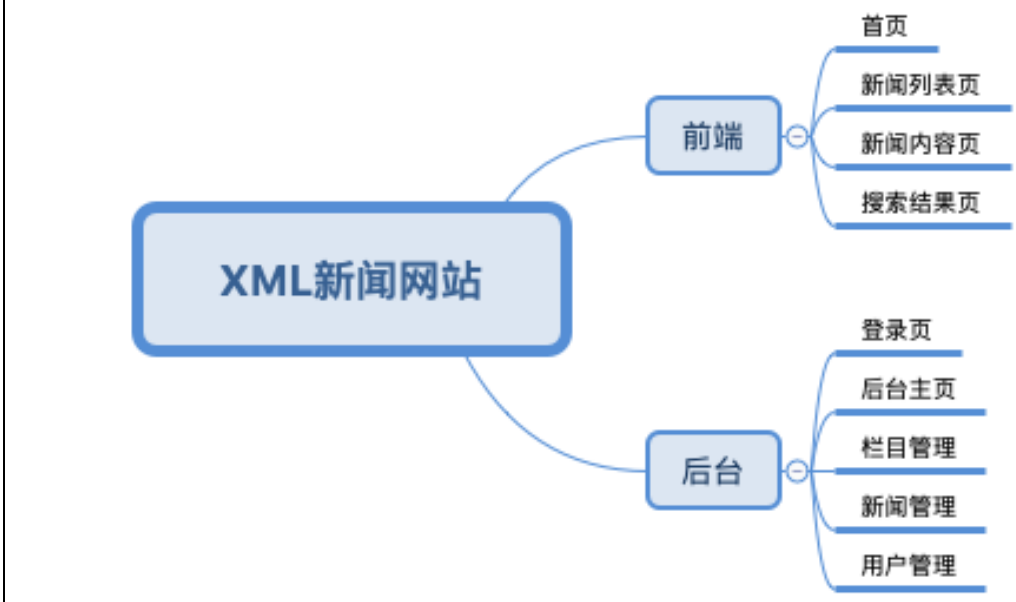
3、设计系统的流程是哪些？

第二部分：实验过程与结果分析

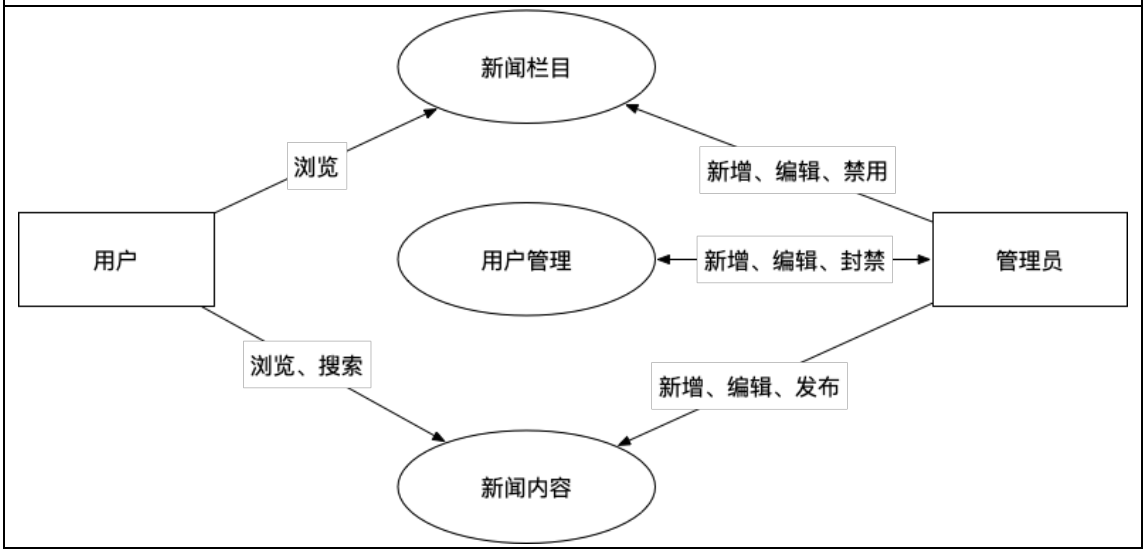
1 功能需求分析

1.1 用例图及思维导图

思维导图



用例图



1.2 用户角度

从用户角度，用户可以查看到 XML 新闻网站的前端页面，前端页面主要包括以下界面：新闻首页、新闻列表页、新闻内容页、搜索结果页。

前端所有页面都包含以下部分，一、主体部分：该部分的内容根据每个页面的功能不同会有所差别。二、搜索框：可以通过输入关键词对新闻标题、新闻内容进行关键词搜索，并跳转到对应的搜索结果页。三、新闻栏目列表：可以在新闻栏目列表中看到目前展示的所有新闻栏目以及每个栏目下对应的文章数，且栏目是按照权重从大到小进行排序展示的。四、顶栏与底栏，顶栏包括 XML 新闻网站的前端链接和后台链接。底栏包括版权信息。

新闻首页的主体内容是新闻列表，用户进入新闻首页可以看到目前发布的所有新闻内容，所有新闻每条新闻内容都包含新闻标题、新闻作者、新闻来源、新闻发布时间、新闻浏览量、新闻所属栏目名称、新闻摘要等数据。

新闻列表页和新闻首页结构基本一致，主体部分上需要显示当前所在的栏目名称，主体部分的新闻内容列表也是该栏目下的所有新闻数据。

新闻内容页的主体部分是该篇新闻的详细内容，其中包括新闻标题、新闻作者、新闻来源、新闻发布时间、新闻浏览量、新闻所属栏目名称、新闻摘要、新闻详细内容。

搜索结果页的主体部分是所有符合关键词的新闻列表数据。

1.3 管理员角度

从管理员角度，管理员可以使用 XML 新闻网站后台查看并管理目前的所有新闻栏目、新闻内容、新闻阅读情况、后台用户。后台包括后台首页、新闻栏目管理模块、新闻内容管理模块、后台用户管理模块。

后台所有页面都包含以下部分，一、主体部分：该部分的内容根据每个页面的功能不同会有所差别。二、侧栏：侧栏中可以查看后台的所有管理模块，包括控制台（后台首页）、栏目管理、新闻管理、用户管理；可以在侧栏直接跳转到各个管理模块。三、顶栏：顶栏包括 XML 新闻网站前端的跳转链接、用户个人管理（编辑个人信息、注销登录）

栏目管理是对新闻栏目的管理模块，包含三个主要功能：展示栏目、新增栏目、编辑栏目。在展示页可以看到的信息为所有新闻栏目的栏目 id、栏目名称、栏目权重、栏目状态。在新增栏目功能中可以设置栏目的名称、权重、状态属性。在编辑栏目功能中可以设置栏目的名称、权重、状态属性。

新闻管理是对新闻内容的管理模块，包含三个主要功能：展示新闻、新增新闻、编辑新闻。在展示页可以看到的信息是所有新闻内容的新闻编号、所属栏目名称、新闻标题、新闻作者、新闻浏览量、新闻发布状态、新闻发布时间。新增新闻可以设置新闻的所属新闻栏目、新闻标题、新闻发布状态、新闻摘要、新闻主体内容。其中新闻主体内容为富文本编辑，可以保存新闻的编辑格式。编辑新闻可以设置新闻的所属新闻栏目、新闻标题、新闻发布状态、新闻摘要。

用户管理是对 XML 新闻网站后台的所有管理员的账号管理。包含两个主要功能：展示所有后台用户、新增后台用户。在展示页可以看到后台用户的用户名、真实姓名、当前状态；并可以通过点击封禁/恢复的操作按钮来更新后台用户的状态，当用户状态为封禁状态时，则无法登陆后台。在新增后台用户功能中可以设置后台用户的用户名、真实姓名、密码、用户状态（默认状态为正常）。

用户个人管理中的编辑个人信息功能，提供对后台个人信息的编辑功能，可以编辑自己的密码、真实姓名。

1.4 系统辅助功能

系统主要功能包括后台用户管理模块、新闻栏目管理模块、新闻内容管理模块。除了这几个核心模块还有对新闻浏览的记录模块，这部分功能模块作为辅助性的功能模块。浏览记录模块记录每个新闻被访问的时间，可以根据这个数据获取到总浏览量、当日浏览量、每篇新闻分别的浏览量。

2 系统详细设计

2.1 前端

前端所有页面都包含以下部分，一、主体部分：该部分的内容根据每个页面的功能不同会有所差别。二、搜索框：可以通过输入关键词对新闻标题、新闻内容进行关键词搜索，并跳转到对应的搜索结果页。三、新闻栏目列表：可以在新闻栏目列表中看到目前展示的所有新闻栏目以及每个栏目下对应的文章数，且栏目是按照权重从大到小进行排序展示的。四、顶栏与底栏，顶栏包括 XML 新闻网站的前端链接和后台链接。底栏包括版权信息。

2.1.1 首页

前端首页主体展示所有状态为发布状态的新闻，新闻的排列方式按照发布时间，最近发布的新闻在上方，发布越久的新闻在后面。新闻

按照列表形式，最下面放置分页器，一页默认 5 篇新闻。

在新闻列表中，新闻每条新闻内容都包含新闻标题、新闻作者、新闻来源、新闻发布时间、新闻浏览量、新闻所属栏目名称、新闻摘要等数据。其中，点击新闻所属栏目名称可以跳转到对应的新闻栏目页面中。点击新闻标题或继续阅读按钮可以跳转到该新闻的内容页查看新闻的正文内容。

2.1.2 新闻列表页

前端新闻列表页主体内容是状态为正常展示状态的栏目下的已发布的新闻列表。在新闻列表的上方显示当前所在的列表名称，新闻列表的内容和首页中的新闻列表样式一致。不同的是属于专有栏目的新闻数据。

2.1.3 新闻内容页

新闻内容也主体内容是展示新闻的新闻标题、新闻作者、新闻来源、新闻发布时间、新闻浏览量、新闻所属栏目名称、新闻摘要、新闻正文内容。在点击进入新闻内容页时会触发对该新闻的浏览记录，记录新闻的编号、浏览的时间。

2.1.4 搜索结果页

搜索结果页的主体内容是展示标题或者新闻正文内容包含搜索关键词的新闻列表，新闻列表的内容和首页中的新闻列表样式一致。

2.2 后台

后台所有页面都包含以下部分，一、主体部分：该部分的内容根据每个页面的功能不同会有所差别。二、侧栏：侧栏中可以查看后台的所有管理模块，包括控制台（后台首页）、栏目管理、新闻管理、用户管理；可以在侧栏直接跳转到各个管理模块。三、顶栏：顶栏包括 XML 新闻网站前端的跳转链接、用户个人管理（编辑个人信息、注销登录）

2.2.1 首页

后台首页主要是重点数据的展示和管理模块的导航入口。首页展示日浏览量、总浏览量、新闻总数、后台用户总数信息。首页包括顶栏、侧栏。顶栏可以进入到 XML 新闻网站的前端首页、可以注销后台登录状态、可以编辑自己的一些信息（密码、真实姓名）

2.2.2 新闻栏目管理

新闻栏目管理包括展示栏目、新增栏目、编辑栏目三大功能。栏目可以对新闻数据进行分类管理，其中栏目具有栏目名称、栏目权重、栏目状态等属性，一篇新闻只可以属于一个栏目。栏目的权重决定栏目在展示时的顺序，权重越高的栏目，展示的顺序越靠前。栏目的状态代表着栏目是否展示，如果栏目的状态为正常，则代表栏目可以正常的显示，如果栏目的状态为禁用，则在前端页面无法查看到该栏目。新增栏目功能可以编辑栏目的名称、权重、状态三个信息。其中权重数据必须是数字。编辑栏目功能可以编辑栏目的名称、权重、状态三个信息。栏目一经创建原则上不可进行删除。

2.2.3 新闻内容管理

新闻内容管理包括展示新闻列表、新增新闻、编辑新闻三大功能。新闻是新闻网站的核心也是主体内容。一篇新闻需要包含以下信息：新闻编号、新闻标题、新闻摘要、新闻所属栏目名称、新闻主体内容、新闻浏览量、新闻作者、新闻发布状态、新闻发布时间。

2.2.4 用户管理

用户管理包括对用户信息的展示，以及新增后台用户。用户管理是通过调用接口方法来操作 XML 文件来实现对后台用户数据的展示和编辑。在展示页可以看到后台用户的用户名、真实姓名、当前状态；并可以通过点击封禁/恢复的操作按钮来更新后台用户的状态，当用户状态为封禁状态时，则无法登陆后台。在新增后台用户功能中可以设置后台用户的用户名、真实姓名、密码、用户状态。

3 系统数据实现

3.1 xml 数据实现

本系统使用 XML 文件来管理后台用户登录，并通过读取和写入 XML 文件来达到对后台用户的查看、添加、编辑功能。

3.1.1 用户信息管理

通过对数据的分析可以得出后台用户需要的字段包括：用户名、密码、真实姓名、用户状态。其中用户名不能有重复的情况出现，不然在登录的时候容易产生问题。用户状态分为正常和封禁两种状态，封禁状态下的用户不能正常登录后台。

XML 数据结构如下：

user.xml

```
<?xml version="1.0" encoding="utf-8"?>
<user_list>
    <user>
        <username>xzfff</username>
        <password>123</password>
        <realname>谢泽丰</realname>
        <status>0</status>
    </user>
    <user>
        <username>zcf</username>
        <password>123</password>
        <realname>郑成锋</realname>
        <status>0</status>
    </user>
</user_list>
```

3.2 sql 数据实现

在 XML 新闻系统中，不可避免的要生成或存储一些持久性数据，如新闻类别、新闻内容等。这些数据按传统方式一般是存储到数据库表中。

3.2.1 新闻栏目管理

新闻栏目管理数据表的字段包括栏目 id、栏目名称、栏目权重、栏目状态。其中栏目权重在新闻前端页面中起到决定展示顺序的作用，权重越大的栏目，顺序越在前面。

新闻栏目数据表结构如下：

lists.sql

lists

表注释： 新闻栏目表

字段	类型	空	默认	注释
id *(主键)*	int(11)	否		栏目id
name	varchar(255)	否		栏目名称
sort	int(11)	否		栏目权重，从大到小排序
is_show	tinyint(1)	否		

索引

键名	类型	唯一	紧凑	字段	基数	排序规则	空	注释
PRIMARY	BTREE	是	否	id	4	A	否	

3.2.2 新闻内容管理

新闻内容管理数据表的字段包括新闻 id、所属栏目 id、标题、作者、新闻摘要、新闻正文、发布时间、是否发布。其中是否发布中 0 代表待发布，1 代表已发布。新闻的发布时间在每次更新新闻数据的时候会随之更新。在前端页面展示时会按照发布时间从近到远进行排序。新闻内容数据表结构如下：

news.sql

news

表注释： 新闻表

字段	类型	空	默认	注释
id *(主键)*	int(11)	否		新闻id
lists_id	int(11)	否		所属栏目id
title	varchar(255)	否		标题
author	varchar(64)	否		作者
summary	text	否		摘要
content	text	否		内容
publish_time	datetime	否		发布时间，排序从近到远
is_show	tinyint(1)	否		是否发布

索引

键名	类型	唯一	紧凑	字段	基数	排序规则	空	注释
PRIMARY	BTREE	是	否	id	20	A	否	

3.2.3 新闻浏览情况管理

新闻浏览情况数据表的字段包括浏览 id、新闻 id、浏览时间。可以通过对新闻浏览表的操作来查询到当天所有新闻的浏览情况以及每篇新闻的浏览情况甚至是统计所有新闻的总浏览量。可以实现的查询有非常多样。

新闻浏览情况数据表结构如下：

views.sql

views

表注释： 浏览表

字段	类型	空	默认	注释
id *(主键)*	int(11)	否		
news_id	int(11)	否		
view_time	datetime	否		

索引

键名	类型	唯一	紧凑	字段	基数	排序规则	空	注释
PRIMARY	BTREE	是	否	id	86	A	否	

4 系统功能实现

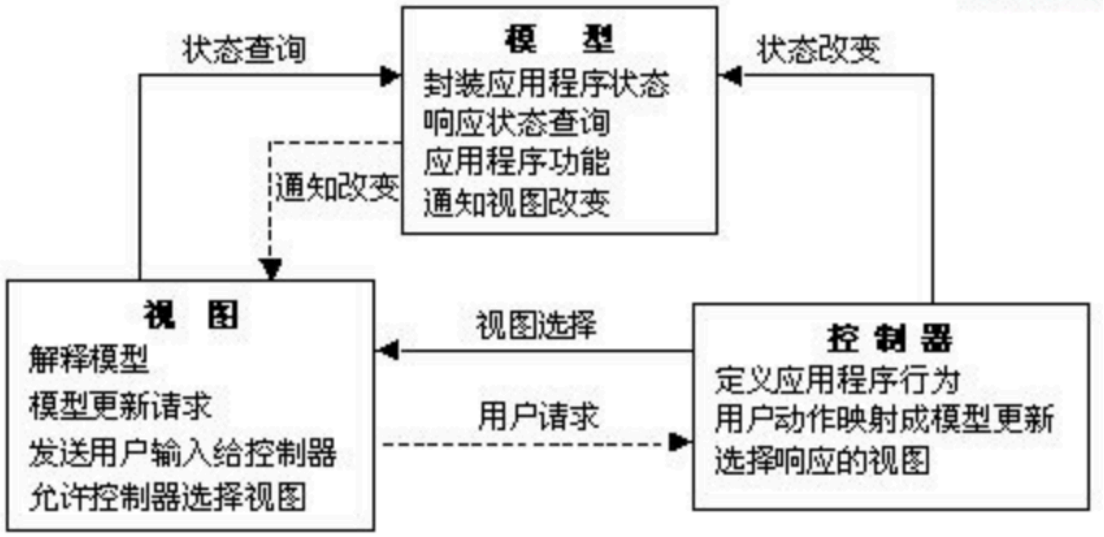
系统基于 ThinkPHP5.1 框架进行开发，使用 MVC 架构来开发业务。

4.1 前端

4.1.1 MVC 架构

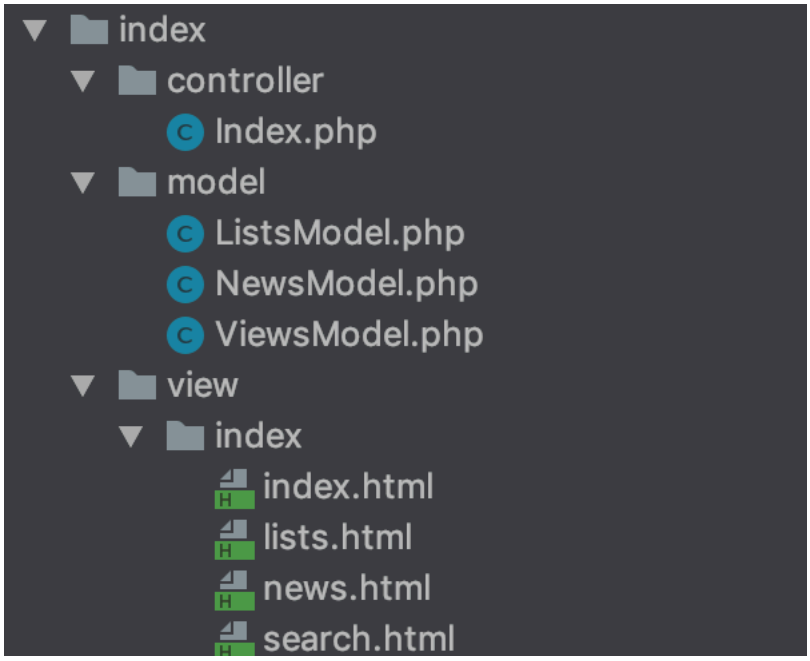
MVC 是模型(model)－视图(view)－控制器(controller)的缩写。用一种业务逻辑、数据、界面显示分离的方法组织代码，将业务逻辑聚集到一个部件里面，在改进和个性化定制界面及用户交互的同时，不需要重新编写业务逻辑。

MVC 架构



实际的业务逻辑中可以通过文件结构来看出系统整体的设计思路

前端代码文件结构

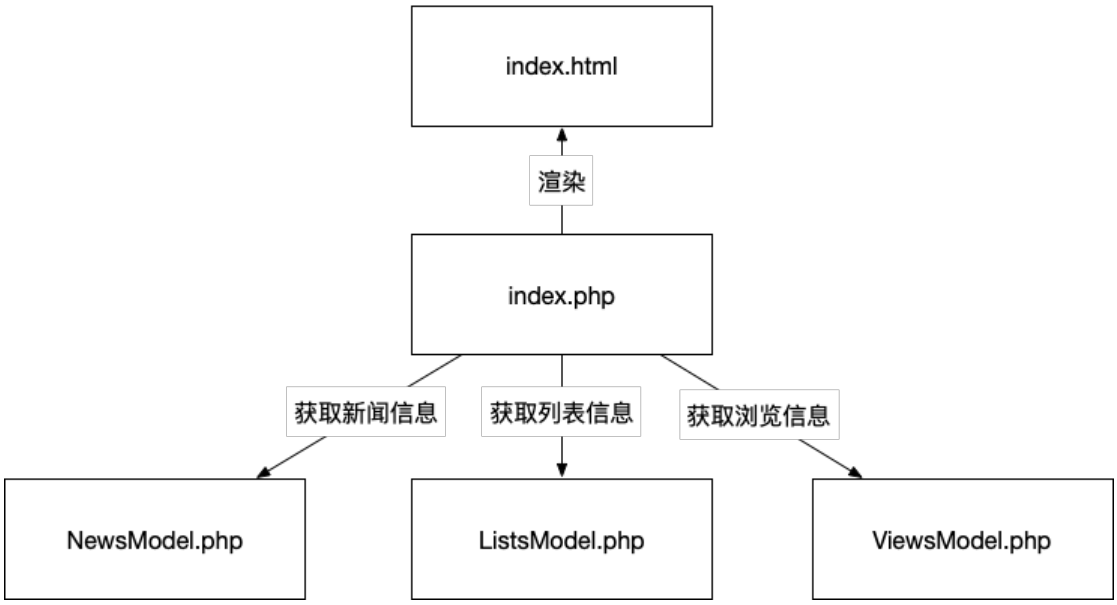


4.1.2 首页实现

首页由以下文件来构成：

index.html/index.php/ListModel.php/NewsModel.php/ViewsModel.php

页面渲染结构



其中 NewsModel.php 是专门用于操作跟 news 数据表相关的操作的，ListsModel.php 和 ViewsModel.php 同理。index.php 作为控制器，在接到浏览器对应的访问后，先到三个 model 层获取首页需要的数据，再渲染到 index.html 中。

此处以 index.php 中的 index 函数为例，来说明前端页面的渲染逻辑

index.php/index

```
    public function index()
    {
        $list_rows = 5;
        $lists_model = new ListsModel();
        $news_model = new NewsModel();
        $views_model = new ViewsModel();
        try {
            // 栏目列表
            $lists_data = $lists_model->where(['is_show' => 0])->order('sort',
'desc')->select();
            // 查询栏目下的新闻数
            foreach ($lists_data as $key => $value) {
                $lists_id = $value['id'];
                $lists_data[$key]['news_num']                                    =
count($news_model->where(['lists_id' => $lists_id, 'is_show' => 1])->select());
            }
            $this->assign('lists_data', $lists_data);

            // 新闻列表
            $news_data        =        $news_model->where(['is_show'        =>
1])->paginate($list_rows);
```

```
            foreach ($news_data->items() as $key => $value) {
                $lists_id = $value['lists_id'];
                $news_data->items()[$key]['lists_name']                          =
$lists_model->field('name')->where(['id' => $lists_id])->find()['name'];
                $news_id = $value['id'];
                $news_data->items()[$key]['views']                               =
$views_model->get_news_views($news_id)['data'];
            }
            $news_page = $news_data->render();
            $this->assign('news_data', $news_data);
            $this->assign('news_page', $news_page);

        } catch (\Exception $e) {
            return data_return(CODE_ERROR, '数据获取失败', $e->getMessage());
        }
        return $this->fetch();
    }
```

从代码中可以看到，代码中分别取出了正常状态的栏目信息用于渲染栏目列表，所有发布状态的新闻信息并通过 for 循环来分别取出新闻的所属栏目名称信息以及新闻浏览数信息，最后栏目信息和新闻信息都作为参数传入到 index.html 中用于页面数据的渲染。

4.1.3 新闻列表页实现

新闻列表页的实现与首页的实现逻辑类似，在此处仅展示实现的核心代码

index.php/lists

```
    public function lists()
    {
        $now_lists_id = input('get.lists_id');
        $list_rows = 5;
        $lists_model = new ListsModel();
        $news_model = new NewsModel();
        $views_model = new ViewsModel();
        try {
            // 栏目名称
            $lists_name    =    $lists_model->field('name')->where(['id'    =>
$now_lists_id])->find()['name'];
            $this->assign('lists_name', $lists_name);

            // 栏目列表
            $lists_data = $lists_model->where(['is_show' => 0])->order('sort',
'desc')->select();
            // 查询栏目下的新闻数
            foreach ($lists_data as $key => $value) {
                $lists_id = $value['id'];
                $lists_data[$key]['news_num']                                    =
count($news_model->where(['lists_id' => $lists_id, 'is_show' => 1])->select());
```

```
            }
            $this->assign('lists_data', $lists_data);

            // 新闻列表
            $news_data = $news_model->where(['is_show' => 1, 'lists_id' =>
$now_lists_id])
                ->paginate($list_rows, false, [
                    'query' => request()->param()
                ]);
            foreach ($news_data->items() as $key => $value) {
                $lists_id = $value['lists_id'];
                $news_data->items()[$key]['lists_name']                    =
$lists_model->field('name')->where(['id' => $lists_id])->find()['name'];
                $news_id = $value['id'];
                $news_data->items()[$key]['views']                         =
$views_model->get_news_views($news_id)['data'];
            }
            $news_page = $news_data->render();
            $this->assign('news_data', $news_data);
            $this->assign('news_page', $news_page);

        } catch (\Exception $e) {
            return data_return(CODE_ERROR, '数据获取失败', $e->getMessage());
        }
        return $this->fetch();
    }
```

4.1.4 新闻内容页实现

新闻内容页的实现与首页的实现逻辑类似，在此处仅展示实现的核心代码

index.php/news

```
    public function news()
    {
        $now_news_id = input('get.news_id');
        $lists_model = new ListsModel();
        $news_model = new NewsModel();
        $views_model = new ViewsModel();
        try {
            // 栏目列表
            $lists_data = $lists_model->where(['is_show' => 0])->order('sort',
'desc')->select();
            // 查询栏目下的新闻数
            foreach ($lists_data as $key => $value) {
                $lists_id = $value['id'];
                $lists_data[$key]['news_num']                              =
count($news_model->where(['lists_id' => $lists_id, 'is_show' => 1])->select());
            }
```

```
            $this->assign('lists_data', $lists_data);

            // 新闻详细信息
            $news_data = $news_model->where(['id' => $now_news_id])->find();
            $lists_id = $news_data['lists_id'];
            $news_data['lists_name'] = $lists_model->field('name')->where(['id'
=> $lists_id])->find()['name'];
            $news_data['views']                                                =
$views_model->get_news_views($now_news_id)['data'];
            $this->assign('news_data', $news_data);

            // 增加一次新闻的 view
            $views_model->set_news_views($now_news_id);
        } catch (\Exception $e) {
            return data_return(CODE_ERROR, '数据获取失败', $e->getMessage());
        }
        return $this->fetch();
    }
```

4.1.5 搜索结果页实现

搜索结果页中需要考虑到关键词的获取，本系统将关键词拼接在 url 的后面，后端使用 GET 方法来获取对应的数据，并通过关键词查出所有相关结果

index.php/search

```
    public function search()
    {
        $keyword = input('get.keyword');
        $lists_model = new ListsModel();
        $news_model = new NewsModel();
        $views_model = new ViewsModel();
        try {
            // 栏目列表
            $lists_data = $lists_model->where(['is_show' => 0])->order('sort',
'desc')->select();
            // 查询栏目下的新闻数
            foreach ($lists_data as $key => $value) {
                $lists_id = $value['id'];
                $lists_data[$key]['news_num']                                  =
count($news_model->where(['lists_id' => $lists_id, 'is_show' => 1])->select());
            }
            $this->assign('lists_data', $lists_data);

            // 搜索结果-新闻列表
            $news_data = $news_model->where('is_show', '=', 1)
                ->where('title|content', 'like', "%{$keyword}%")
                ->select();
```

```
            foreach ($news_data as $key => $value) {
                $lists_id = $value['lists_id'];
                $news_data[$key]['lists_name']                                    =
$lists_model->field('name')->where(['id' => $lists_id])->find()['name'];
                $news_id = $value['id'];
                $news_data[$key]['views']                                         =
$views_model->get_news_views($news_id)['data'];
            }
            $this->assign('news_data', $news_data);

        } catch (\Exception $e) {
            return data_return(CODE_ERROR, '数据获取失败', $e->getMessage());
        }
        return $this->fetch();
    }
```

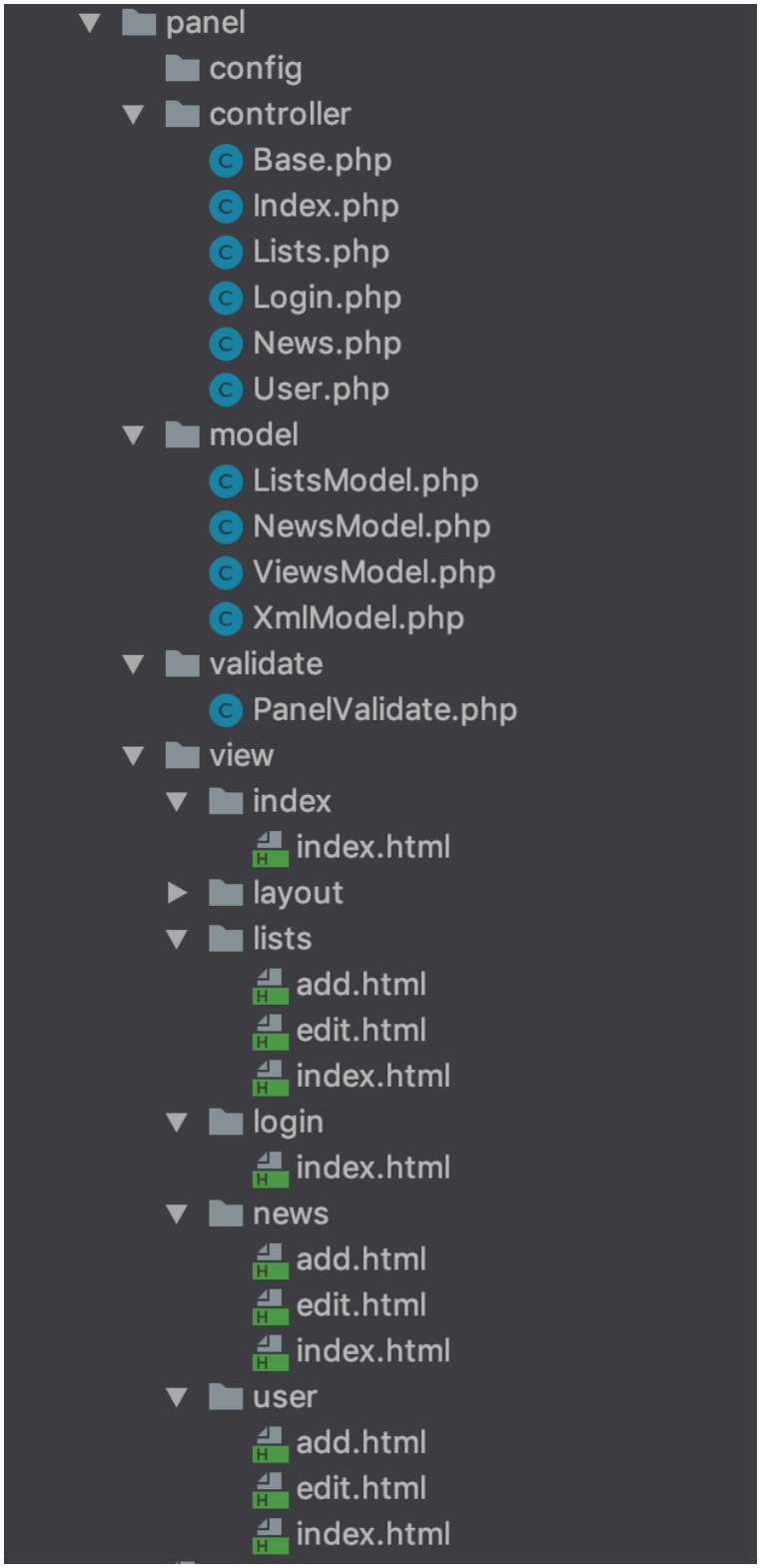
4.2 后台

4.2.1 AJAX 架构

后台中由于要展示很多数据，单纯的使用 MVC 会导致代码的耦合性过高，为了保证代码分模块，分别实现对应的功能模块，将一些数据接口改成了 AJAX 的形式。AJAX 在浏览器与 Web 服务器之间使用异步数据传输（HTTP 请求），这样就可使网页从服务器请求少量的信息，而不是整个页面。

后台的目录结构如下：

后台代码文件结构



4.2.2 登录实现

在进入首页之前需要完成登录操作，因此还有 login 文件来专门对用户登录操作进行校验。用户在 login.html 界面输入账号密码后，通过 do_login 函数对账号密码进行验证（这里是通过操作 xml 文件来验证账号密码的），在通过验证后，后台会将用户的信息记录在 session 中。

同时除了 login 文件，其他所有控制器都会继承 Base 控制器，在 Base 控制器中的初始化函数中，就会对用户的登录状态进行检查，如果用户没有登录信息，则会强制返回到登录界面。

4.2.3 首页实现

首页的实现与前端中 MVC 的实现类似，在 index.php 文件中完成对界面的渲染。这里在开发中为了让代码具有复用性，在后台的 html 文件中，对代码进行了拆分，将顶栏和侧栏的代码拆分到了 layout 下，这样每个页面只需要 include 一下 layout 下的顶栏和侧栏代码文件就可以显示对应样式，减少了很多重复的无用代码量。

4.2.4 新闻栏目管理实现

新闻栏目管理实现除了顶栏与侧栏外，使用 dataTables 对栏目数据进行展示，并提供了新增栏目功能和编辑栏目信息功能。

这里面对数据的展示的核心代码如下：

获取所有栏目数据进行展示

```
public function get_lists()
{
    $input_data = input('post.aoData');
    $aoData = json_decode($input_data);
    $lists_model = new ListsModel();
    $offset = 0;
    $limit = 10;
    foreach ($aoData as $key => $val) {
        if ($val->name == 'iDisplayStart')
            $offset = $val->value;
        if ($val->name == 'iDisplayLength')
            $limit = $val->value;
    }

    $lists_info = $lists_model->get_lists($offset, $limit);
    // 计算总数
    $lists_info_total = $lists_model->get_lists(0, 0);
    $resp['recordsTotal'] = count($lists_info_total['data']);
    $resp['recordsFiltered'] = count($lists_info_total['data']);
    $resp['data'] = $lists_info['data'];
    foreach ($lists_info['data'] as $key => $value) {
        if ($value['is_show'] == 0) {
            $value['is_show'] = '正常';
        } else {
            $value['is_show'] = '禁用';
        }
    }
    echo json_encode($resp);
```

```
    }
```

4.2.5 新闻内容管理实现

新闻内容的管理与新闻栏目的管理类似，但是新闻内容的编辑一般是带有样式的。因此在新闻正文的编辑中使用的是富文本编辑器。本系统采用的解决方案是 UEditor，该编辑器是百度发布的一款富文本编辑器，在样式的调整上非常完善。也包含图片上传、视频上传等功能。

这里就富文本编辑器的调用核心代码进行展示

富文本编辑器 UEditor 编辑器调用

```
<div class="col-lg-12">
    <div class="form-group">
        <label>内容</label>

        <!-- 加载编辑器的容器 -->
        <script id="container" name="news_content" type="text/plain"></script>
        <!-- 配置文件 -->
        <script type="text/javascript"
                src="/xml-news/public/ueditor/ueditor.config.js"></script>
        <!-- 编辑器源码文件 -->
        <script type="text/javascript"
                src="/xml-news/public/ueditor/ueditor.all.js"></script>
        <!-- 实例化编辑器 -->
        <script type="text/javascript">
            var ue = UE.getEditor('container');
        </script>
        <button type="button" class="btn btn-default">提交</button>
    </div>
</div>
```

4.2.6 用户管理实现

用户管理模块中提供了三个主要方法：展示所有用户信息、新增一个用户、编辑用户的状态。用户的管理使用的是 xml 文件来记录用户的信息，通过操作 xml 文件来对用户进行管理。这里面也涉及了很多与 xml 相关的操作，这里不多做描述，在下文会进行展开。

5 系统运行截图

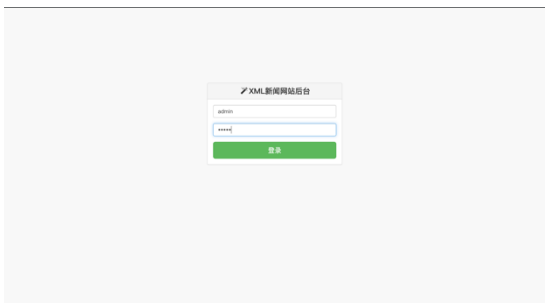
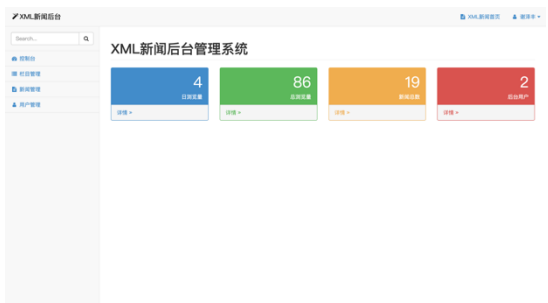
5.1 前端运行截图

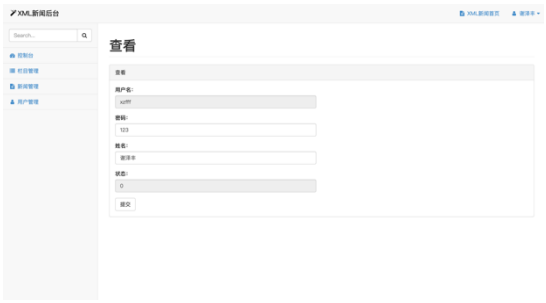
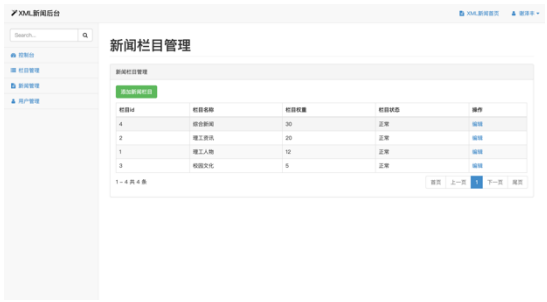
新闻首页 01	新闻首页 02
	

新闻列表页	新闻正文页
	

搜索结果页	
	

5.2 后台运行截图

用户登录页	后台首页
	

后台首页（右上角带列表）	编辑用户信息
	
新闻栏目页	新增一个新闻栏目
	
编辑新闻栏目	新闻列表页
	
新增新闻内容	编辑新闻内容
	
用户管理页	新增后台用户
	

6 系统中 xml 文件及操作 xml 相关代码

6.1 xml 文件结构

xml 文件结构如下：

user.xml

```
<?xml version="1.0" encoding="utf-8"?>
<user_list>
    <user>
        <username>xzfff</username>
        <password>123</password>
        <realname>谢泽丰</realname>
        <status>0</status>
    </user>
    <user>
        <username>zcf</username>
        <password>123</password>
        <realname>郑成锋</realname>
        <status>1</status>
    </user>
</user_list>
```

6.2 操作 xml 相关代码

在操作 xml 中，分为两块，一是对 xml 进行直接操作的核心代码，二是辅助对 xml 操作的一些辅助方法，用于对 xml 数据的一些辅助处理。

操作 xml 的相关代码有以下的一些核心方法：

读取 xml 文件

```
public function read_user_xml()
{
    $filename = $_SERVER['DOCUMENT_ROOT'] . "/xml-news/public/xml/user.xml";

    // 读取文件方法 1:适用于本地文件
    $handle = fopen($filename, 'r');
    $content = '';
    while (false != ($a = fread($handle, 8080))) {//返回 false 表示已经读取到文件末尾
        $content .= $a;
    }
    fclose($handle);

    // 读取文件方法 2:适用于本地文件、远程文件
    /*
    $ctx = stream_context_create(array(
            'http' => array(
                'timeout' => 1 //设置超时
            )
        )
```

```
    );
    $content = file_get_contents($filename, 0, $ctx);
    */

    return $content;
}
```

校验用户登录信息

```
public function find_user_xml($username, $password)
{
    $content = $this->read_user_xml();
    $dom = new \DOMDocument();
    $dom->loadXML($content);
    $root = $dom->documentElement;
    $users = $root->getElementsByTagName('user');
    foreach ($users as $key => $val) {
        $username_xml =
$users->item($key)->getElementsByTagName('username')->item(0)->nodeValue;
        $password_xml =
$users->item($key)->getElementsByTagName('password')->item(0)->nodeValue;
        if ($username_xml == $username && $password_xml == $password) {
            $panel_user['username'] = $username;
            $panel_user['password'] = $password;
            $panel_user['realname'] =
$users->item($key)->getElementsByTagName('realname')->item(0)->nodeValue;
            $panel_user['status'] =
$users->item($key)->getElementsByTagName('status')->item(0)->nodeValue;
            return data_return(CODE_SUCCESS, '登录成功', $panel_user);
        }
    }
    return data_return(CODE_ERROR, '登录失败');
}
```

新增用户前检查用户名是否已经存在

```
public function has_user_xml($username)
{
    $content = $this->read_user_xml();
    $dom = new \DOMDocument();
    $dom->loadXML($content);
    $root = $dom->documentElement;
    $users = $root->getElementsByTagName('user');
    foreach ($users as $key => $val) {
        $username_xml =
$users->item($key)->getElementsByTagName('username')->item(0)->nodeValue;
        if ($username_xml == $username) {
            return data_return(CODE_SUCCESS, '存在用户');
```

```
        }
    }
    return data_return(CODE_ERROR, '不存在用户');
}
```

新增一个用户到 xml 中

```
public function add_user_xml($username, $password, $realname, $status)
{
    $content = $this->read_user_xml();
    $dom = new \DOMDocument();
    $dom->loadXML($content);
    $root = $dom->documentElement;

    $new_user_xml = $dom->createElement('user');
    $root->appendChild($new_user_xml);
    $username_xml = $dom->createElement('username', $username);
    $new_user_xml->appendChild($username_xml);
    $password_xml = $dom->createElement('password', $password);
    $new_user_xml->appendChild($password_xml);
    $realname_xml = $dom->createElement('realname', $realname);
    $new_user_xml->appendChild($realname_xml);
    $status_xml = $dom->createElement('status', $status);
    $new_user_xml->appendChild($status_xml);

    $content = $dom->saveXML();
    return $content;
}
```

编辑用户信息到 xml

```
public function edit_user_xml($username, $status)
{
    $content = $this->read_user_xml();
    $dom = new \DOMDocument();
    $dom->loadXML($content);
    $root = $dom->documentElement;
    $users = $root->getElementsByTagName('user');
    foreach ($users as $key => $val) {
        $username_xml                                                    =
$users->item($key)->getElementsByTagName('username')->item(0)->nodeValue;
        if ($username_xml == $username) {

$users->item($key)->getElementsByTagName('status')->item(0)->nodeValue        =
$status;
            $content = $dom->saveXML();
            return $content;
        }
```

```
    }
    return $content;
}
```

将更新后的 xml 数据写入到 xml 文件中

```
public function write_user_xml($content)
{
    $filename = $_SERVER['DOCUMENT_ROOT'] . "/xml-news/public/xml/user.xml";
    $file = fopen($filename, "w"); // w 代表替换写入
    fwrite($file, $content);
    fclose($file);
}
```

除此之外，对 xml 的操作还有一些辅助函数，这里展示两个比较核心的辅助函数，echo_xml 将 xml 字符串转换成一个对象，便于输出查看 xml 数据。xmlToArray 方法将 xml 字符串转换成一个数组，便于操作。

将 xml 字符串转换成一个对象

```
public function echo_xml($content)
{
    $xml_object = xml($content, 200, []); // 格式化 xml 字符串为对象
    return ($xml_object);
}
```

xml 字符串转换成一个数组

```
//将 XML 转为 array
function xmlToArray($xml)
{
    //禁止引用外部 xml 实体
    libxml_disable_entity_loader(true);
    $values        =        json_decode(json_encode(simplexml_load_string($xml,
'SimpleXMLElement', LIBXML_NOCDATA)), true);
    return $values;
}
```

7 思考-如何优化设计一个系统的功能

我认为要优化设计一个系统的功能，要从以下方面来进行考虑：

1、功能需求分析

我们在进行软件开发时，无论是采用面向对象方法还是传统方法，我认为最首先要做的就是了解需求。由于用例图是从用户角度来描述系统功能的，所以在进行需求分析时，使用用例图可以更好描述系统应具备什么功能。软件建模的其他部分都是从用例图开始的。这些图以每一个参与系统开发的人员都可以理解的方式列举系统的业务需求。

这样我们就可以更加全面的了解系统的业务和需要的功能。

2、找到优化的切入点，删减冗余功能

在优化功能上，我们应该考虑到哪些功能是可以优化的。这就要求我们从使用者的角度来看待这一款产品。用户的需求核心是什么，这也就可以确定我们功能的核心是什么。以此来列举出产品的目标用户群体的需求优先级，以此来确定产品哪些功能是主要的，哪些功能是次要的。以此来确定哪些功能是可以进行优化的。

8 思考-设计系统的注意事项

我认为从所有系统的设计上来说，最应该关注的两个点是：

1、系统设计需要主动而且全面地考虑需求

2、系统设计需要达到资源和需求的平衡

9 思考-设计系统流程

1、问题的定义及规划

此阶段是软件开发与需求放共同讨论，主要确定软件的开发目标及其可行性。

2、需求分析

在确定软件开发可行性的情况下，对软件需要实现的各个功能进行详细需求分析。需求分析阶段是一个很重要的阶段，这一阶段做的好，将为整个软件项目的开发打下良好的基础。“唯一不变的是变化本身”，同样软件需求也是在软件爱你开发过程中不断变化和深入的，因此，必须定制需求变更计划来应付这种变化，以保护整个项目的正常进行。

3、软件设计

此阶段中偶要根据需求分析的结果，对整个软件系统进行设计，如系统框架设计、数据库设计等。软件设计一般分为总体设计和详细设计。还的软件设计将为软件程序编写打下良好的基础。

4、程序编码

此阶段是将软件设计的结果转化为计算机可运行的程序代码。在程序编码中必定要制定统一、符合标准的编写规范。以保证程序的可读性、易维护性。提高程序的运行效率。

5、软件测试

在软件设计完成之后要进行严密的测试，一发现软件在整个软件设计过程中存在的问题并加以纠正。整个测试阶段分为单元测试、组装测试、系统测试三个阶段进行。测试方法主要有白盒测试和黑盒测试。