**福州大学软件学院**

**软件工程本科生**

**《Web程序设计实践》课程报告**

|  |  |
| --- | --- |
|  |  |
| **院 系：** | **计算机与大数据学院(软件学院)** |
| **专 业：** | **软件工程** |
| **年 级：** | **2020级** |
| **学 号：** | **222000331** |
| **姓 名：** | **林玥利** |
|  |  |
|  |  |

2023年01月12日

1. **项目概况**
2. 、项目简介

“LBlog”博客系统分为前台和后台两部分，采用了Yii2.0框架下的Advanced应用模板。后台能实现对文章、评论、管理员、用户的管理。在此基础上，运用AdminLTE-2.3.11模板进行美化、新增dashboard界面、将内容部分引入summernote文本编辑器。前台能实现文章的展示、搜索、标签云查找等基本功能。在此基础上，进行前端美化、增加“关于我”页面。

该博客系统能基本满足系统管理和登入系统查看文章的功能，前端设计比较美观。

1. 、使用的技术和工具

使用VSCode作为编辑器，使用Html、Css、JavaScript和php语言进行开发。

使用Wampserver集成Windows Apache Mysql PHP安装环境。

使用Google Chrome浏览器调式代码。

1. 、为系统增加、扩展所作工作内容
2. 进行了前后台前端美化工作
3. 前台

增加背景图；

将最新回复换成最新文章，直接点击文章名可以跳转到文章详情页；

将文章列表、评论列表、文章详情页面用半透明的圆角边框包括起来，增强视觉效果；

调整部分字体的颜色；

1. 后台

运用AdminLTE-2.3.11模板

修改导航栏（主、侧）内容；

增加背景图；

将大部分组件设置为半透明，增强视觉效果；

将导航栏设定为固定定位；

删除面包屑；

将用户数、管理员数、文章数、评论数呈现在首页；

将评论、文章时间分布和评论文章审核情况做成图表；

1. 后台添加Dashboard界面
2. 展示用户数、管理员数、文章数、评论数

在图标集里面选取符合条件的图标

点击“查看更多”可以跳转到相应页面

1. 用曲线图展示文章发布时间分布情况
2. 用曲线图展示评论发布时间分布情况
3. 用饼图展示文章发布状态
4. 用饼图展示评论审核情况
5. 文章编辑界面增强，引入可视化编辑器；

使用summernote编辑器，为新增文章，修改文章，修改评论的内容部分引入可视化编辑器，用户可以根据需求排版文字、上传图片等。

1. 增加“关于我”页面

其中包含一些自我介绍。

1. 实现RBAC权限管理
2. **系统基础功能的设计与实现**

（一）、基础博客系统架构设计

1、backend和frontend

是前后台入口，相当于一个单独的Basic应用，有自己的mvc目录，配置文件目录，入口文件的目录，其中：

assets是资源文件目录，存放着需要用到的css、js文件地址

config是配置文件目录

controllers、models、views构成项目的mvc，分别存放着项目的控制器、模型已经视图文件。

web是入口文件目录

1. console

是控制台入口，用来存放需要控制台运行的程序，比如放一些定时执行的程序，或者需要在更底层的操作性系统上运行的功能。

1. common

是一个公用的目录，比如一些前后台共有的模型文件就可以放在这里，博客系统里面的文章模型类、评论模型类都会放在这里，前后台公用。

1. environments

放环境的配置文件

1. 、功能及接口的设计
2. 后台

文章的增删查改，其中：

ActiveRecord提供了一套面向对象的接口，利用其提供的find()方法或者findBySql()方法来构建DB查询，利用其提供的insert、delete、update、save、方法实现插入、更新和删除功能；

**文章新增actionCreate**，首先判断用户是否拥有新增文章的操作权限，若有则将客户端写入的内容保存到数据库中，完成保存操作后跳转到新增的文章查看界面，若未成功，则跳转到新增界面；

public function actionCreate()

    {

        $model = new Post();

        if ($model->load(Yii::$app->request->post()) && $model->save()) {

            return $this->redirect(['view', 'id' => $model->id]);

        } else {

            return $this->render('create', [

                'model' => $model,

            ]);

        }

    }

**文章删除actionDelete**，判断是否有权限，若有则从数据库删除该文章并跳转页面至index。

 public function actionDelete($id)

    {

        $this->findModel($id)->delete();

        return $this->redirect(['index']);

    }

**文章查看actionView**调用了findModel函数，从Post类中回传文章进行展示；

public function actionView($id)

    {

        return $this->render('view', [

            'model' => $this->findModel($id),

        ]);

    }

**文章修改actionUpdates**，首先判断用户是否拥有修改文章的操作权限，若有则先找到文章，获取到file，完成修改操作后再提交修改；

public function actionUpdate($id)

    {

        $model = $this->findModel($id);

        if ($model->load(Yii::$app->request->post()) && $model->save()) {

            return $this->redirect(['view', 'id' => $model->id]);

        } else {

            return $this->render('update', [

                'model' => $model,

            ]);

        }

    }

评论的删查改，其中：

**评论删除actionDelete**，调用findModel函数判断用户是否有操作权限，需提交数据库修改，成功则返回index页面。

public function actionDelete($id)

    {

        $this->findModel($id)->delete();

        return $this->redirect(['index']);

    }

**评论修改actionUpdate**，调用findModel函数判断用户是否有操作权限，需提交数据库修改，成功则返回评论具体界面，失败则返回修改界面；

public function actionUpdate($id)

    {

        $model = $this->findModel($id);

        if ($model->load(Yii::$app->request->post()) && $model->save()) {

            return $this->redirect(['view', 'id' => $model->id]);

        } else {

            return $this->render('update', [

                'model' => $model,

            ]);

        }

    }

**评论查看actionView**，调用了findModel函数，从Comment类中回传评论进行展示；

public function actionView($id)

    {

        return $this->render('view', [

            'model' => $this->findModel($id),

        ]);

    }

**评论的审核actionApprove**,调用findModel函数判断用户是否有操作权限，调用approve()方法审核。

public function actionApprove($id)

    {

        $model = $this->findModel($id);

        if($model->approve())  //审核

        {

            return $this->redirect(['index']);

        }

    }

**用户管理actionView**，传递searchModel和dataProvider两个对象数组至index

public function actionView($id)

    {

        return $this->render('view', [

            'model' => $this->findModel($id),

        ]);

    }

**用户创建actionCreate**

public function actionCreate()

    {

        $model = new User();

        if ($model->load(Yii::$app->request->post()) && $model->save()) {

            return $this->redirect(['view', 'id' => $model->id]);

        } else {

            return $this->render('create', [

                'model' => $model,

            ]);

        }

    }

**用户修改actionUpdate**

public function actionUpdate($id)

    {

        $model = $this->findModel($id);

        if ($model->load(Yii::$app->request->post()) && $model->save()) {

            return $this->redirect(['index']);

        } else {

            return $this->render('update', [

                'model' => $model,

            ]);

        }

    }

**用户删除actionDelete**

public function actionDelete($id)

    {

        $this->findModel($id)->delete();

        return $this->redirect(['index']);

    }

**用户登录认证：**

用户组件 yii\web\User 用来管理用户的认证状态；

配置⽤户组件 yii\web\User

指定⼀个含有实际认证逻辑的认证类: app\models\User

实现 yii\web\IdentityInterface 接⼝

+ fifindIdentity()

+ fifindIdentityByAccessToken()

- getId()

- getAuthKey()

- validateAuthKey()

使⽤⽤户组件 yii\web\User

• $identity = Yii::$app->user->identity; 检测当前⽤户身份

• $id = Yii::$app->user->id; 当前⽤户的ID。 未认证⽤户则为 Null

• $isGuest = Yii::$app->user->isGuest; 判断当前⽤户是否是游客

• Yii::$app->user->login($identity); 将当前⽤户的身份登记到 yii\web\User。根据设置，⽤session或cookie记录⽤户身份， ⽤户的认证状态将在整个会话中得以维持。

• Yii::$app->user->logout(); 注销⽤户：

**前后台认证的分离**：

1、把认证类改为：common\models\Adminuser

2、初步调整Adminuser验证类

增加字段

Gii修改代码

拷贝user的认证代码

3、建立后台登录的表单模型类AdminLoginForm

4、调试验证类代码，去掉status\_active常量

5、调整英文提示

6、设定后台专用的session，和前台分开

1. 前台

**数据小部件ListView**:

⽤于显示数据提供者DataProvider提供的数据; 每条数据⽤⼀个指定的⼦视图⽂件来渲染，可以 在⼦视图中⾃由的布置⻚⾯; 由于有dataprovider提供数据，⾃然就集成了分⻚、排序这些特性，可以⽅便地写出显示数据以及管理数据的界⾯。

<?= ListView::widget([

                'id'=>'postList',

                'dataProvider'=>$dataProvider,

                'itemView'=>'\_listitem',//子视图,显示一篇文章的标题等内容.

                'layout'=>'{items} {pager}',

                'pager'=>[

                        'maxButtonCount'=>8,

                        'nextPageLabel'=>Yii::t('app','下一页'),

                        'prevPageLabel'=>Yii::t('app','上一页'),

        ],

        ])?>

**给文章增加URL属性：**

每篇⽂章都有⼀个对应的链接，可以通过 调⽤urlManager的createUrl⽅法，附上⽂章的id等参数，从⽽得到对应⽂章的链接。⽤getUrl给⽂章增加了url属性的好处：

1. 不⽤到处调⽤createUrl⽅法

2. 当链接发⽣变化时，修改很⽅便

3. URL美化时，会更⽅便

public function getUrl()

    {

        return Yii::$app->urlManager->createUrl(

            ['post/detail','id'=>$this->id,'title'=>$this->title]);

    }

**标签云的实现：**

分两个步骤实现：首先要在标签类中写一个findTagsWeights静态方法，这个方法用来生成标签云需要的数据。接着创建一个自定义小部件，用来渲染第一步生成的数据，得到标签云。

**URL美化：**

设置Apache的配置文件：设置虚拟主机名、虚拟主机的web根目录、打开重写引擎、如果请求的文件或目录不存在就重写、重写的规则是把请求转给index.php去处理

配置UrlManager组件：开启url美化，去掉‘index.php?r=’、如果开启脚本名会显示‘index.php’、实现伪静态、设置文章详细页面的美化规则、设置文章列表页面的梅花规则、后台文章和评论的列表页美化、后台资料详细页美化、后台增删改页面的美化。

1. 、测试验证模块的正确性、部署、运行

根据视频教程搭建完成的博客系统能正常运行，完成基本功能。

1. **系统扩展功能的设计与实现**

**（一）、进行了前后台前端美化工作**

1、前台

**增加背景图：**

将图片保存在css文件目录下，并在site.css文件中设置body的样式为：

background-image: url(./background7.jpg);

    background-size: cover;

**将最新回复换成最新文章**，直接点击文章名可以跳转到文章详情页：

首先要在文章类中写一个findRecentPosts静态方法，这个方法用来生成最新文章需要的数据。接着创建一个自定义小部件RecentPostWidget，用来渲染第一步生成的数据，得到最新文章小部件。

public static function findRecentPosts($limit=5)

    {

        return Post::find()->where(['status'=>2])->orderBy('create\_time DESC')

        ->limit($limit)->all();

    }

public function run()

    {

        $postString='';

        foreach ($this->recentPosts as $post)

        {

            $postString.='<div class="post-list">'.

                    '<div class="title">'.

                    '<p style="font-size:12pt;color:bule">

                    《<a href="'.$post->url.'">'.Html::encode($post->title).'</a>》</p>'.

                    '<hr></div></div>';

        }

        return  $postString;

    }

将文章列表、评论列表、文章详情页面用半透明的**圆角边框**包括起来，增强视觉效果：

在site.css文件中设置.post样式

.post{

    background-color: rgb(255, 255, 255, 0.5);

    padding: 10px 20px;

    border-radius: 20px;

    border: 2px solid rgb(255, 255, 255, 0.4);

    margin-bottom: 20px;

}

调整部分字体的颜色：设置color

2、后台

**运用AdminLTE-2.3.11模板**：

<div class="container" style="padding-top: 50px;">

        <?= $content ?>

</div>

**修改导航栏（主、侧）内容：**

根据AdminLTE-2.3.11模板修改设置header标签和aside标签中的内容，主导航为登录和退出登录，侧导航为首页、文章管理、评论管理、用户管理和管理员管理。

**增加背景图：**

（同后台）

**将大部分组件设置为半透明，增强视觉效果：**

background-color:rgb(... , ... , ... , 透明度）；

**将导航栏设定为固定定位：**

position:fixed;

再调整四个方向上的距离

**删除面包屑：**

**（二）后台添加Dashboard界面**

**1、展示用户数、管理员数、文章数、评论数**



将模板文件中的相关代买复制到site.php文件的内容区域；

将需要用到的css样式表文件和js脚本文件路径写入到Appassets文件中；

用php代码从数据库获取用户数、管理员数、文章数、评论数相关数据；

$user\_count = User::find()->where(['>','id','0'])->count();

$adminuser\_count = Adminuser::find()->where(['>','id','0'])->count();

$comment\_count = Comment::find()->where(['>','id','0'])->count();

$post\_count = Post::find()->where(['>','id','0'])->count();

将取到的数据渲染到组件上；

<div class="inner">

    <h3><?=$post\_count?></h3>

<p>总文章数</p>

</div>

点击“查看更多”可以跳转到相应页面；

**2、用曲线图展示文章发布时间分布情况**



将模板文件中的相关代买复制到site.php文件的内容区域；

将需要用到的css样式表文件和js脚本文件路径写入到Appassets文件中；

设置一个数组$data\_post存放每个月的文章发布数量；

$date\_post = array(0,0,0,0,0,0,0,0,0,0,0,0);

从数据库里取出文章发布时间的时间戳，运用intval函数将时间戳转换成月份，再将月份作为数组的索引值累加，得到一个data\_post数组；

$post\_time = Post::findAll(['status'=>2,]);

foreach($post\_time as $items){

  $month\_post = intval(date("m",$items->create\_time));

  $date\_post[$month\_post-1] = $date\_post[$month\_post-1]+1;

}

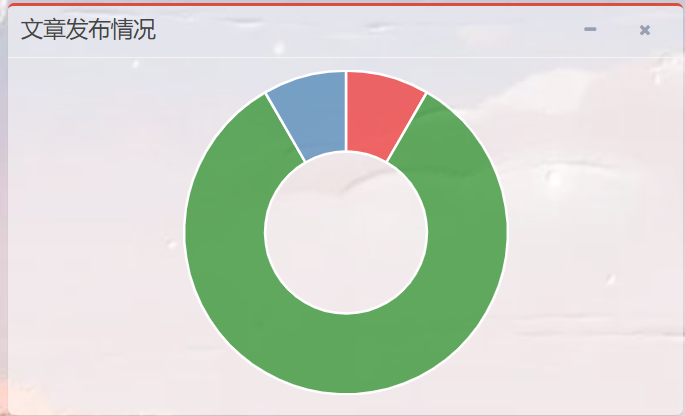
将$date\_post渲染到js编写的曲线图上

data: ['<?php echo $date\_post[0] ?>', '<?php echo $date\_post[1] ?>', '<?php echo $date\_post[2] ?>', '<?php echo $date\_post[3] ?>', '<?php echo $date\_post[4] ?>', '<?php echo $date\_post[5] ?>', '<?php echo $date\_post[6] ?>', '<?php echo $date\_post[7] ?>','<?php echo $date\_post[8] ?>', '<?php echo $date\_post[9] ?>', '<?php echo $date\_post[10] ?>', '<?php echo $date\_post[11] ?>'],

**3、用曲线图展示评论发布时间分布情况**

（同上）

**4、用饼图展示文章发布状态**



将模板文件中的相关代买复制到site.php文件的内容区域；

将需要用到的css样式表文件和js脚本文件路径写入到Appassets文件中；

设置三个变量$post\_draft、$post\_done、$post\_guidang分别表示文章草稿、文章已发布和文章已归档；从数据库中查询处于这三种状态的文章个数

$post\_draft = Post::find()->where(['=','status','1'])->count();

$post\_done = Post::find()->where(['=','status','2'])->count();

$post\_guidang = Post::find()->where(['=','status','3'])->count();

将得到的数据渲染到饼图上

var PieData = [

      {

        value: <?php echo $post\_draft ?>,

        color: "rgb(238,44,44,0.7)",

        highlight: "rgb(238,44,44,0.8)",

        label: "草稿"

      },

      {

        value: <?php echo $post\_done ?>,

        color: "rgb(34,139,34,0.7)",

        highlight: "rgb(34,139,34,0.8)",

        label: "已发布"

      },

      {

        value: <?php echo $post\_guidang ?>,

        color: "rgb(70,130,180,0.7)",

        highlight: "rgb(70,130,180,0.8)",

        label: "已归档"

      },

    ];

**5、用饼图展示评论审核情况**

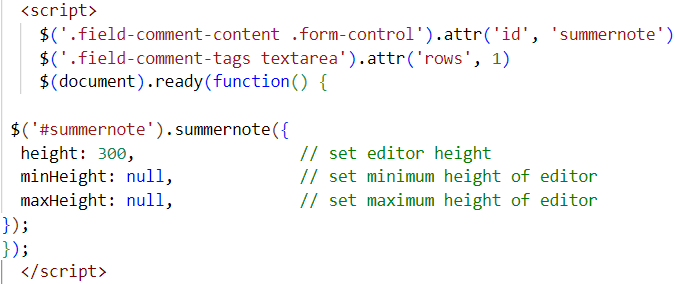
（同上）

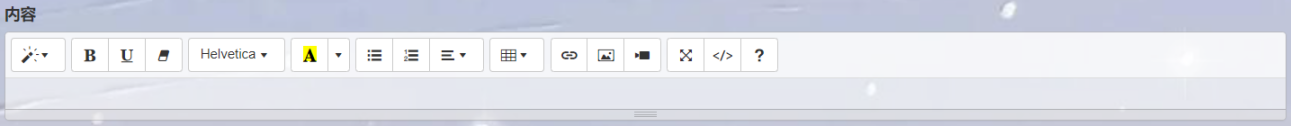
**（三）、文章编辑界面增强，引入可视化编辑器；**

使用summernote编辑器，为新增文章，修改文章，修改评论的内容部分引入可视化编辑器，用户可以根据需求排版文字、上传图片等。

将相关css和js文件路径添加到AppAssets.php文件中，然后在需要添加可视化编辑器的页面添加js代码引入编辑器；

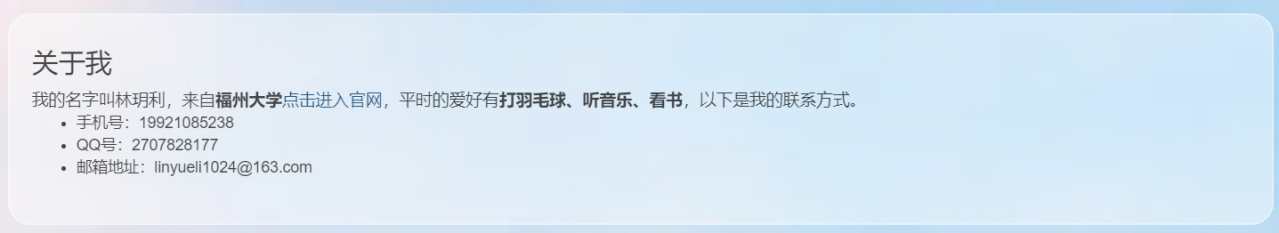
为需要添加可视化编辑器的标签设置id=“summernote”；





**（四）、增加“关于我”页面**

其中包含一些自我介绍。



**（五）、实现RBAC权限管理**

系统的权限分为文章管理、文章操作、评论审核，系统管理包括前面的三种权限；

文章管理员可以对文章进行创建、修改、删除操作；

文章操作员只能对文章进行删除操作，不能对文章进行创建和修改操作；

评论审核员能对评论进行审核；

登录系统后在管理员界面可以设置管理员的权限，对应的管理员登陆系统依据设置好的权限对系统进行操作；

**1、存取控制过滤器：**

是⼀种通过yii\fifilters\AccessControl类来实现的简单授权⽅法，⾮常适⽤于仅需要简单的存取控制的应⽤。

在控制器中behaviors⽅法中设定存取的规则。当用户请求⼀个动作的时候，ACF会检查存取规则，判断该用户是否被允许执⾏所请求的动作。

**2、基于角色的存取控制RBAC：**

第一步：建立数据授权

1.1 配置应用组件authManager为yii\rbac\DbManager

1.2 用数据迁移来创建四张数据库表，准备来存放授权数据

1.3 通过 authManager 提供的 API 创建一个控制台命令初始化授权数据

第二步：执行权限检查

if(!Yii::$app->user->can('createPost')){

            throw new ForbiddenHttpException('对不起，您没有进行操作的权限。');

        }

1. **思考题**
2. Web开发中MVC模式的应用有什么好处，为什么要做这样的划分？（掌握Yii框架中控制器，模型，视图的用途和使用方法）

耦合性低:视图层和业务层分离，这样就允许更改视图层代码而不用重新编译模型和控制器代码，同样，一个应用的业务流程或者业务规则的改变只需要改动MVC的模型层即可。因为模型与控制器和视图相分离，所以很容易改变应用程序的数据层和业务规则。

重用性高:MVC模式允许使用各种不同样式的视图来访问同一个服务器端的代码，因为多个视图能共享一个模型，它包括任何WEB（HTTP）浏览器或者无线浏览器（wap）。

部署快，生命周期成本低 MVC使开发和维护用户接口的技术含量降低。使用MVC模式使开发时间得到相当大的缩减，它使程序员集中精力于业务逻辑，界面程序员（HTML和JSP开发人员）集中精力于表现形式上。

可维护性高 分离视图层和业务逻辑层也使得WEB应用更易于维护和修改。

1. 什么是约定优于配置（convention over configuration）？Yii框架中哪些地方体现了约定优于配置设计范式？

约定优于配置，也称作按约定编程，是一种软件设计范式，旨在减少软件开发人员需做决定的数量，获得简单的好处，而又不失灵活性。

Yii框架中的一个model模型即对应一个数据库表，同时，一个model也对应几个同样命名的控制器，在前台和后台的代码中含有对应控制器命名的视图文件夹，文件夹内再是其他视图的名称。又例如 Yii 中对于 action 通配符的规范，XXXController.php中有一个名为 actionEdit() 的方法，那么体现在网页 URL 上，就是是除去 action 前缀的动作函数名。

1. 什么是 ORM？什么是ActiveRecord模式？

ORM 是 Object Relational Mapping 的缩写，译为“对象关系映射”，它解决了对象和关系型数据库之间的数据交互问题。

ActiveRecord 也属于 ORM 层，由 Rails 最早提出，遵循标准的 ORM 模型：表映射到记录，记录映射到对象，字段映射到对象属性。配合遵循的命名和配置惯例，能够很大程度的快速实现模型的操作，而且简洁易懂。

ActiveRecord 的主要思想是：

1. 每一个数据库表对应创建一个类，类的每一个对象实例对应于数据库中表的一行记录；通常表的每个字段在类中都有相应的 Field ；

2. ActiveRecord 同时负责把自己持久化，在 ActiveRecord 中封装了对数据库的访问，即 CURD; ；

3. ActiveRecord 是一种领域模型 (Domain Model) ，封装了部分业务逻辑；

ActiveRecord 比较适用于：

1. 业务逻辑比较简单，当你的类基本上和数据库中的表一一对应时 , ActiveRecord 是非常方便的，即你的业务逻辑大多数是对单表操作；

2. 当发生跨表的操作时 , 往往会配合使用事务脚本 (Transaction Script) ，把跨表事务提升到事务脚本中；

3. ActiveRecord 最大优点是简单 , 直观。 一个类就包括了数据访问和业务逻辑 . 如果配合代码生成器使用就更方便了；

这些优点使 ActiveRecord 特别适合 WEB 快速开发。

1. Gii代码生成器有哪些优缺点？如何进行代码模板的定制？

优点：

* 适合分布式开发，强调个体。
* 公共服务器压力和数据量都不会太大。
* 速度快、灵活。
* 任意两个开发者之间可以很容易的解决冲突。
* 离线工作。

缺点：

* 资料少（起码中文资料很少）。
* 学习周期相对而言比较长。
* 不符合常规思维。
* 代码保密性差，一旦开发者把整个库克隆下来就可以完全公开所有代码和版本信息。

1. 使用查询构建器QueryBuilder而不是使用手工编写原始SQL语句有什么好处？Yii中这种查询构建器的写法称为方法链（Method Chaining）。在众多现代语言/框架中你可以看到它的身影，如 Java, Python, jQuery等。这种写法有什么优点？

查询构建器也是建⽴在 DAO 基础之上，可让你创建程序化的、DBMS无关的SQL语句，并且，这样创建的SQL语句，⽐原⽣的SQL语句更易读、更安全。

方法链的一个好处，是可以减少你使用对象名的次数。调用的方法越多，能够减少的次数就越多。因此，这个方法也能一定程度上减少需要阅读、测试、调试、维护的代码数量。这个好处不大，但也是有用的。

1. 让调用过程更接近自然语言。

2. 把原本参数列表复杂的方法化作多个参数列表简单的方法来使用。

3. 减少不必要的代码量。

这个三点都是有益于开发的，所以方法链的存在很有意义。

1. 用户身份认证与RBAC中，角色在用户和权限之间起到了什么作用？

RBAC（基于角色的访问控制），就是用户通过角色与权限进行关联。简单地说，一个用户拥有若干角色，每一个角色拥有若干权限。这样，就构造成“用户-角色-权限”的授权模型。在这种模型中，用户与角色之间，角色与权限之间，一般者是多对多的关系。

1. 前端基于Bootstrap这样的框架开发与自己从零开始编写CSS相比有什么好处？

bootstrap是一款国际流行的前端框架，方便快捷地进行响应式布局和站点总体样式构建是它作为css框架最明显的优点。Bootstrap是一个非常规范和完善的框架，在我学习的道路上经常听到大牛前端说的话就是“阅读优秀的代码是进步的途径”，Bootstrap就是这样的优秀代码。此外还有以下优点：节省时间、定制化、设计方面的因素、一致性、保持持续更新、易于集成、响应式、对未来技术具有兼容性、竞争力、详尽的说明文档。

1. URL 美化有什么好处？其实现机制是什么？

好处：

提高安全性，可以有效的避免一些参数名、ID等完全暴露在用户面前，如果用户随便乱输的话，不符合规则的话直接会返回个404或错误页面，这比直接返回500或一大堆服务器错误信息要好的多。

美化URL，去除了后缀名、参数串等，可以自己组织精简更能反映访问模块内容的URL。  
 更有利于搜索引擎的收入，通过对URL的一些优化，可以使搜索引擎更好的识别与收录网站的信息。

实现机制：

开启Apache的mod\_rewrite 模块，修改 Apache 的 AllowOverride，在与index.php文件同级目录（web目录)下添加文件“.htaccess”，配置应用的urlManager。通过修改Apache的配置，和urlManage实现URL的美化。

1. 什么是缓存，Web应用中缓存有什么用？Yii框架支持哪些类型缓存？其实现机制分别是什么？

Web缓存是指一个Web资源（如html页面，图片，js，数据等）存在于Web服务器和客户端（浏览器）之间的副本。缓存会根据进来的请求保存输出内容的副本；当下一个请求来到的时候，如果是相同的URL，缓存会根据缓存机制决定是直接使用副本响应访问请求，还是向源服务器再次发送请求。

使用Web缓存的作用：

减少网络带宽消耗

无论对于网站运营者或者用户，带宽都代表着金钱，过多的带宽消耗，只会便宜了网络运营商。当Web缓存副本被使用时，只会产生极小的网络流量，可以有效的降低运营成本。

降低服务器压力

给网络资源设定有效期之后，用户可以重复使用本地的缓存，减少对源服务器的请求，间接降低服务器的压力。同时，搜索引擎的爬虫机器人也能根据过期机制降低爬取的频率，也能有效降低服务器的压力。

减少网络延迟，加快页面打开速度

带宽对于个人网站运营者来说是十分重要，而对于大型的互联网公司来说，可能有时因为钱多而真的不在乎。那Web缓存还有作用吗？答案是肯定的，对于最终用户，缓存的使用能够明显加快页面打开速度，达到更好的体验。

在yii中缓存一共分为四种，分别为数据缓存,片段缓存,页面缓存,HTTP缓存

数据缓存是指将一些 PHP 变量存储到缓存中，使用时再从缓存中取回。 它也是更高级缓存特性的基础，例如[查询缓存](https://www.yiichina.com/doc/guide/2.0/caching-data" \l "query-caching) 和[内容缓存](https://www.yiichina.com/doc/guide/2.0/caching-content)。

如下代码是一个典型的数据缓存使用模式。 其中 $cache 指向[缓存组件](https://www.yiichina.com/doc/guide/2.0/caching-data" \l "cache-components)：

// 尝试从缓存中取回 $data $data = $cache->get($key);

**if** ($data === **false**) {

// $data 在缓存中没有找到，则重新计算它的值

// 将 $data 存放到缓存供下次使用

$cache->set($key, $data);

}

// 这儿 $data 可以使用了。

**片段缓存**指的是缓存页面内容中的某个片段。例如，一个页面显示了逐年销售额的摘要表格， 可以把表格缓存下来，以消除每次请求都要重新生成表格的耗时。 片段缓存是基于[数据缓存](https://www.yiichina.com/doc/guide/2.0/caching-data)实现的。

在[视图](https://www.yiichina.com/doc/guide/2.0/structure-views)中使用以下结构启用片段缓存：

**if** (**$this**->beginCache($id)) {

// ... 在此生成内容 ...

**$this**->endCache();

}

调用 [beginCache()](https://www.yiichina.com/doc/api/2.0/yii-base-view" \l "beginCache()-detail) 和 [endcache()](https://www.yiichina.com/doc/api/2.0/yii-base-view" \l "endCache()-detail) 方法包裹内容生成逻辑。 如果缓存中存在该内容，[beginCache()](https://www.yiichina.com/doc/api/2.0/yii-base-view" \l "beginCache()-detail) 方法将渲染内容并返回 false， 因此将跳过内容生成逻辑。否则，内容生成逻辑被执行， 一直执行到[endCache()](https://www.yiichina.com/doc/api/2.0/yii-base-view" \l "endCache()-detail) 时， 生成的内容将被捕获并存储在缓存中。

和[数据缓存](https://www.yiichina.com/doc/guide/2.0/caching-data)一样，每个片段缓存也需要全局唯一的 $id 标记。

**页面缓存**指的是在服务器端缓存整个页面的内容。 随后当同一个页面被请求时，内容将从缓存中取出，而不是重新生成。

页面缓存由 [yii\filters\PageCache](https://www.yiichina.com/doc/api/2.0/yii-filters-pagecache) 类提供支持，该类是一个[过滤器](https://www.yiichina.com/doc/guide/2.0/structure-filters)。 它可以像这样在控制器类中使用：

**public** **function** **behaviors**(){

**return** [

[

'class' => 'yii\filters\PageCache',

'only' => ['index'],

'duration' => 60,

'variations' => [

\Yii::$app->language,

],

'dependency' => [

'class' => 'yii\caching\DbDependency',

'sql' => 'SELECT COUNT(\*) FROM post',

],

],

];

}

HTTP缓存：Web 应用还可以利用客户端缓存 去节省相同页面内容的生成和传输时间。

通过配置 [yii\filters\HttpCache](https://www.yiichina.com/doc/api/2.0/yii-filters-httpcache) 过滤器，控制器操作渲染的内容就能缓存在客户端。 [HttpCache](https://www.yiichina.com/doc/api/2.0/yii-filters-httpcache) 过滤器仅对 GET 和 HEAD 请求生效， 它能为这些请求设置三种与缓存有关的 HTTP 头。

[Last-Modified](https://www.yiichina.com/doc/api/2.0/yii-filters-httpcache" \l "$lastModified-detail)

[Etag](https://www.yiichina.com/doc/api/2.0/yii-filters-httpcache" \l "$etagSeed-detail)

[Cache-Control](https://www.yiichina.com/doc/api/2.0/yii-filters-httpcache" \l "$cacheControlHeader-detail)

1. **实践心得体会**

本次的实践作业是建立在《魏曦教你学Yii2.0》教程的源代码基础上增加扩展功能，不断完善的。刚开始下载源代码发现无法运行，会时不时出现报错。因为刚开始不了解Yii框架，所以一直报错令我非常苦恼，就利用空闲时间和同学们配合着视频一起讨论研究。后面基本上将视频完整看一遍后发现原先的代码有几处需要修改，修改后可以成功运行。看完一边视频后，也对Yii框架有了整体的了解，为后面添加扩展功能打下了基础。

添加扩展功能的时候，因为刚开始还没有将项目中各个文件的作用彻底弄清楚，所以一开始有些手足无措。于是，又去看了视频教程中对各文件作用的讲解，比如模型文件、控制器文件、布局文件、入口文件等等。掌握了这些才能开始开发扩展功能。

首先开发的是后台的美化，因为有学过一些web开发，所以这部分不是很困难。从git-hub上下载了老师推荐的模板套入到后台的main.php文件中，并使用浏览器的调试工具对样式和内容进行修改。

接下来开发的是dashboard界面，用可视化图表来展示系统的总体情况。因为模板中有包含dashboard,所以开发起来相对容易。

接为系统的内容部分添加了可视化编辑器summernote，这一部分也不是很困难，只要跟着官网上的指导来操作即可。

然后开发了“关于我”界面，这一部分只需要编写html文件即可。

最后完善了RBAC功能。一开始我觉得这个功能我是无法实现的，在网上也搜索了大量的资料发现非常复杂。但后面整理报告的时候又发现视频中是有这部分的讲解的，只需要在控制器文件中的静态方法中添加权限判断。经过观看视频和调试，最终也完成了这部分的开发。

但还有一些老师推荐的扩展功能，比如文章分类、附件管理等没时间或是没有能力完成。我相信通过今后的不断学习，我一定会有足够的能力完成这些扩展功能。

完成了这次的实践作业之后再去回想这段时间，我从完全不会Yii框架、完全不懂php语言到现在基本掌握Yii框架，还能在此基础之上完成扩展功能的开发，我也看到了自己是具备一定实践能力的。除此之外，我还和同学们一起讨论研究，体会到了沟通交流在开发中起到了至关重要的作用。

这次的实践作业是学期末考试复习的间隙不断学习完善的，虽然很累，但是在完成之后看到自己的成果是还是很激动的。在今后的学习中，我会不断完善自己，勇于挑战自己，争取在下一次的实践作业中能有更好的表现。