



南开大学
Nankai University

“小组” 小组_实现文档

1. Gii生成博客系统

- gii是yii中的一个模块。可以通过配置应用yii\base\Application::modules属性开启它，gii脚手架可以帮助我们自动生成模型，控制器，crud的一些操作等，gii被设计成盖度可定制和可扩张的代码生成工具。使用它可以大幅提高应用开发速度

本博客系统的基础框架也是借助gii来构建的，下面进行具体过程的介绍。

1.1 进入gii工具

将网页路由改为r=gii即可进入gii工具。

1.2 生成模型文件

这一步是根据数据库中的数据表生成model文件的过程，下面的图示以post这张表为例，其余表的生成模型方法完全一致。

Model Generator

CRUD Generator

Controller Generator

Form Generator

Module Generator

Extension Generator

Model Generator

This generator generates an ActiveRecord class for the specified database table.

Table Name

post

Model Class Name

Post

☐ Standardize Capitals

☐ Singularize

Namespace

app\models

Base Class

yii\db\ActiveRecord

Database Connection ID

db

☐ Use Table Prefix

Generate Relations

All relations

☒ Generate Relations from Current Schema

☐ Generate Labels from DB Comments

☐ Generate ActiveRecord

☐ Enable 118N

☒ Use Schema Name

Code Template

default (F:\xampp-low\htdocs\yii222\vendor\yiisoft\yii2-gii\src\generators\model\def...

Preview

填写时需注意的一点是模型文件的首字母一般需要大写，按照图片填写完内容之后点击preview，然后点击generate文件即可生成。按照上述方法依次生其余数据库表的模型类即可。

1.3 生成模型类增删改查的类

这一步是根据上一步生成的模型类来生成对应的增删改查类的过程，下面的图示以Post这个类为例，其余类的生成增删改查类的方法完全一致。

CRUD Generator

This generator generates a controller and views that implement CRUD (Create, Read, Update, Delete) operations for the specified data model.

Model Class

Post

Search Model Class

common\models\PostSearch

Controller Class

backend\controllers\PostController

View Path

@app/views/post

Base Controller Class

yii\web\Controller

Widget Used in Index Page

GridView

☐ Enable I18N

☐ Enable Pjax

Code Template

default (F:\xampp-low\htdocs\yii222\vendor\yiisoft\yii2-gii\src\generators\crud\defa...

Preview

填写时需注意的一点由于我们在后台进行增删改查操作，所以我们将控制器类只需要生成在backend文件夹下即可，按照图片填写完内容之后点击preview，然后点击generate文件即可生成。按照上述方法依次生其余类的增删改查类即可。

至此，我们就完成了gii生成博客系统的过程了，可以看到我们的博客系统已经初具雏形了。

2. 内容页面

2.1 查看页面

本博客系统中的查看页面的功能主要是在上一步中借助gii工具生成的，我们不需要添加额外的代码。

2.2 修改新增页面

2.2.1 下拉菜单，数组助手类

使用gii生成的框架还有一些不足，比如在我们新建文章的时候需要自行输入作者的ID等信息，这是我们所不能接受的，我们理想中新建一篇文章时输入作者信息时应有如下两个特点：

- 应输入作者的名字而非ID
- 应该使用下拉菜单来限制用户的输入

为了实现这两个要求，我们使用了yii框架中的ArrayHelper类来帮助我们实现。

```
1 $psObjs = Poststatus::find()->all();
2 $allStatus = ArrayHelper::map($psObjs, 'id', 'name');
```

首先我们用find()方法来获取所有的状态信息，然后我们用ArrayHelper类的静态方法map来得到键值对，这样我们就获得了下拉菜单所需要的数据格式。

```
1 $form->field($model, 'status')->dropDownList($allStatus, ['prompt'=>'请选择状态']);
```

然后我们就可以用dropDownList()方法来创建一个下拉菜单，值得注意的是，这只是实现这个功能的其中一种方法，在本博客系统中我们实际使用的是下面2.2.2节中介绍的另一种方法。

2.2.2 查询构建器

- 查询构建器

查询构建器是建立在 DAO 基础之上，可创建程序化的、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。

在yii\backend\views\news_form.php及yii\backend\views\post_form.php中，我们需要添加文章查询的功能，通过查询构建器可以创建SQL语句。在news_from.php中：

```
1 $allStatus = (new \yii\db\Query())
2 ->select(['name', 'id'])
3 ->from('newsstatus')
4 ->indexBy('id')
5 ->column();
```

类似地，在post中进行查询：

```
1 $allStatus = (new \yii\db\Query())
2 ->select(['name', 'id'])
3 ->from('poststatus')
4 ->indexBy('id')
5 ->column();
```

上面的查询语句能够将以id为索引的新闻或博客以列表的形式呈现出来。

2.2.3 ActiveRecord方法

同样在common文件夹下的model中，需要对Post.php以及News.php完善内容页面修改后各参数值的更新，如标签、修改时间等。

可能发生的情况如下：

- 修改内容后，在保存之前状态的更新
- 修改内容后，查询之后标签的更新
- 修改并保存内容后，标签的更新
- 删除后，对应标签数量的更新

由于post和news的操作类似，对应的代码实现如下：

```
1 public function beforeSave($insert)
2 {
3     if(parent::beforeSave($insert))
4     {
5         if($insert)
6         {
7             $this->create_time = time();
8             $this->update_time = time();
9         }
10        else
11        {
12            $this->update_time = time();
13        }
14
15        return true;
16    }
17    else
18    {
19        return false;
20    }
21 }
22 }
23
24 public function afterFind()
25 {
26     parent::afterFind();
27     $this->_oldTags = $this->tags;
28 }
29
30 public function afterSave($insert, $changedAttributes)
31 {
32     parent::afterSave($insert, $changedAttributes);
33     Tag::updateFrequency($this->_oldTags, $this->tags);
34 }
```

```

35
36 public function afterDelete()
37 {
38     parent::afterDelete();
39     Tag::updateFrequency($this->tags, '');
40 }

```

2.2.4 Tag的处理

在上面已经提到了文章进行增删改时会涉及到标签的处理，在这一节进行具体的实现。

- 标签增加
- 标签内容或个数修改

这些在Tag.php中进行具体实现。

```

1 public static function addTags($tags)
2 {
3     if(empty($tags)) return ;
4
5     foreach ($tags as $name)
6     {
7         $aTag = Tag::find()->where(['name'=>$name])->one();
8         $aTagCount = Tag::find()->where(['name'=>$name])->count();
9
10        if(!$aTagCount)
11        {
12            $tag = new Tag;
13            $tag->name = $name;
14            $tag->frequency = 1;
15            $tag->save();
16        }
17        else
18        {
19            $aTag->frequency += 1;
20            $aTag->save();
21        }
22    }
23 }
24
25 public static function removeTags($tags)
26 {
27     if(empty($tags)) return ;
28
29     foreach($tags as $name)
30     {

```

```

31     $aTag = Tag::find()->where(['name'=>$name])->one();
32     $aTagCount = Tag::find()->where(['name'=>$name])->count();
33
34     if($aTagCount)
35     {
36         if($aTagCount && $aTag->frequency<=1)
37         {
38             $aTag->delete();
39         }
40         else
41         {
42             $aTag->frequency -= 1;
43             $aTag->save();
44         }
45     }
46 }
47 }
48
49 public static function updateFrequency($oldTags,$newTags)
50 {
51     if(!empty($oldTags) || !empty($newTags))
52     {
53         $oldTagsArray = self::string2array($oldTags);
54         $newTagsArray = self::string2array($newTags);
55
56         self::addTags(array_values(array_diff($newTagsArray,$oldTagsArray)));
57         self::removeTags(array_values(array_diff($oldTagsArray,$newTagsArray)));
58     }
59 }
60
61
62 public static function findTagWeights($limit=20)
63 {
64     $tag_size_level = 5;
65
66     $models=Tag::find()->orderBy('frequency desc')->limit($limit)->all();
67     $total=Tag::find()->limit($limit)->count();
68
69     $stepper=ceil($total/$tag_size_level);
70
71     $tags=array();
72     $counter=1;
73
74     if($total>0)
75     {
76         foreach ($models as $model)
77         {

```

```

78         $weight=ceil($counter/$stepper)+1;
79         $tags[$model->name]=$weight;
80         $counter++;
81     }
82     ksort($tags);
83 }
84 return $tags;
85 }

```

其中个别地方需要进行边界条件的处理。

上述代码中用到了 `string2array` 及 `array2string` 这两个函数：

```

1 public static function string2array($tags)
2 {
3     return preg_split('/\s*,\s*/',trim($tags),-1,PREG_SPLIT_NO_EMPTY);
4 }
5
6 public static function array2string($tags)
7 {
8     return implode(', ', $tags);
9 }

```

主要是在进行文章修改时，可能将“a,b,c”的标签改成“a,c,d”的标签，我们需要先求出修改前后的差集，将新增的进行添加，删去的进行减少。这一操作需要用到 `array_diff` 方法，所以要将string类的tag转换成数组类再求差集。

2.2.5 GridView

这一步需要在backend\views\post\index.php和backend\views\news\index.php中进行修改：

```

1 <?= GridView::widget([
2     'dataProvider' => $dataProvider,
3     'filterModel' => $searchModel,
4     'columns' => [
5         // ['class' => 'yii\grid\SerialColumn'],
6
7         //'id',
8         ['attribute'=>'id',
9         'contentOptions'=>['width'=>'30px'],
10     ],
11     'title',
12     //'author_id',
13     ['attribute'=>'authorName',

```



```

14     'label'=>'作者',
15     'value'=>'author.nickname',
16 ],
17 // 'content:ntext',
18     'tags:ntext',
19     //'status',
20     ['attribute'=>'status',
21     'value'=>'status0.name',
22     'filter'=>Poststatus::find()
23         ->select(['name','id'])
24         ->orderBy('position')
25         ->indexBy('id')
26         ->column(),
27 ],
28     //'create_time:datetime',
29     //'update_time:datetime',
30     ['attribute'=>'update_time',
31     'format'=>['date','php:Y-m-d H:i:s'],
32 ],
33
34
35     ['class' => 'yii\grid\ActionColumn'],
36 ],
37 ])); ?>

```

在文章管理页面，就能更清晰直观的展现各文章的属性信息。

2.2.6 DataProvider

在PostSearch以及NewsSearch的php文件中，为实现分页管理并安装id进行排序，在search函数中添加以下代码：

```

1  $dataProvider = new ActiveDataProvider([
2      'query' => $query,
3      'pagination' => ['pageSize'=>10],
4      'sort'=>[
5          'defaultOrder'=>[
6              'id'=>SORT_DESC,
7          ],
8          //'attributes'=>['id','title'],
9      ],
10 ]]);

```

2.2.3 Search类

- 属性和搜索表单对应
- 数据规则要重写，使其符合表单提交的这种验证需求
- 搜索的实现，是靠查询构建器来程序化地构建查询，然后交给数据提供者，在后续阶段去执行查询
- 排序的实现，是程序化地设置数据提供者DataProvider的sort配置来实现的

在进行内容页面管理时，需要构建一个继承自Post的PostSearch和继承自News的NewsSearch类来进行数据库的查询。为了实现在搜索框内可以输入作者名称进行查询，我们在类中使用

`attributes()` 函数添加了一个作者名的属性：

```
1 public function attributes()  
2 {  
3     return array_merge(parent::attributes(), ['authorName']);  
4 }
```

由于可以根据作者名顺序或倒序排序，在函数中添加：

```
1 $dataProvider->sort->attributes['authorName'] =  
2 [  
3     'asc'=>['Adminuser.nickname'=>SORT_ASC],  
4     'desc'=>['Adminuser.nickname'=>SORT_DESC],  
5 ];
```

3. 后端搭建

使用Yii框架中的Gii模块生成各数据表的Controller、model和增删改查view之后，为符合我们的后端需求，现做如下修改：

3.1 评论视图页面

3.1.1 评论index页面-截取显示前10字符

为保证评论界面整齐，将超过10字符的内容只截取显示前10字符，使用Getter、Setter定义属性的方法：

在model中创建getBeginning()函数，函数内容如下：

```
1 public function getBeginning()
```

```

2      {
3          $tmpStr = strip_tags($this->content);
4          $tmpLen = mb_strlen($tmpStr);
5          //若tmpLen>10, 则额外显示...
6          return mb_substr($tmpStr,0,10,'utf-8').((($tmpLen>10)?'...':'');
7
8      }

```

在GridView中修改显示内容如下：

```

1  [
2      'attribute'=>'content',
3      'label'=>'内容',
4      'value'=>'beginning',
5  ],

```

3.1.2 设置status显示及搜索筛选功能

1.将显示的值由status值改为status表中对应名称（User字段同理）：

```

1  'value'=>'status0.name',
2  //getStatus已经由Gii生成在model中
3  public function getStatus0()
4  {
5      return $this->hasOne(Commentstatus::className(), ['id' => 'status']);
6  }

```

2.设置GridView搜索框为下拉筛选：

```

1  'filter'=>Commentstatus::find()
2      ->select(['name','id'])
3      ->orderBy('position')
4      ->indexBy('id')
5      ->column(),

```

3.如果status值为1（对应为未审核），将其显示样式改为红色。

```

1  [

```

```

2         'contentOptions'=>
3             function($model)
4             {
5                 return ($model->status==1)?['class'=>'bg-danger']:[];
6             }
7     ],

```

3.1.3 评论审核功能

1.重写gridview最后一栏中的ActionColumn，添加动作按钮approve

```

1         'class' => 'yii\grid\ActionColumn',
2         'template'=>'{view} {update} {delete} {approve}',
3

```

2.添加approve按钮匿名函数，弹出选择框

```

1     'buttons'=>
2         [
3             'approve'=>function($url,$model,$key)
4             {
5                 $options=[
6                     'title'=>Yii::t('yii', '审核'),
7                     'aria-label'=>Yii::t('yii', '审核'),
8                     'data-confirm'=>Yii::t('yii', '你确定通过这条评
9                     'data-method'=>'post',
10                    'data-pjax'=>'0',
11                    ];
12                return Html::a('<span class="glyphicon glyphicon
13
14                },
15        ],

```

3.在controller中添加点击该按钮后的动作

```

1     public function actionApprove($id)
2     {
3         $model = $this->findModel($id);
4         $model->approve(); //审核
5         return $this->redirect(['index']);

```

```
6     }
```

4.在model中进行数据库内容更新

```
1 public function approve()  
2     {  
3         $this->status = 2; //设置评论状态为已审核  
4         return ($this->save()?true:false);  
5     }
```

3.2 用户/管理员界面

3.2.1 前后端登录认证分离

在后端config/main.php中设置后台专用的session字段，与前端区别。修改后端认证类，使其在AdminUser中查找用户信息并验证。

```
1 'session' => [  
2 // this is the name of the session cookie used for login on the backend  
3     'name' => 'advanced-backend',  
4 ],
```

3.2.2 管理员重置密码功能

1.重写gridview最后一栏中的ActionColumn，添加动作按钮resetpwd

```
1 ['class' => 'yii\grid\ActionColumn',  
2     'template'=>'{view} {update} {resetpwd}',  
3 ],
```

2.添加approve按钮匿名函数，弹出修改密码界面

```

1  'buttons'=>[
2      'resetpwd'=>function($url,$model,$key)
3          {
4              $options=[
5                  'title'=>Yii::t('yii','重置密码'),
6                  'aria-label'=>Yii::t('yii','重置密码'),
7                  'data-pjax'=>'0',
8              ];
9              return Html::a('<span class="glyphicon glyphicon'
10          },
11 ],

```

3.新建model: resetpwdForm.php, 进行数据库内容更新;

```

1  public function resetPassword($id)
2      {
3          if (!$this->validate()) {
4              return null;
5          }
6
7          $admuser = Adminuser::findOne($id);
8          $admuser->setPassword($this->password);
9          $admuser->removePasswordResetToken();
10
11          return $admuser->save() ? true : false;
12      }

```

4.在controller中添加点击该按钮后的动作, 确认后返回主界面;

```

1  public function actionResetpwd($id)
2      {
3          $model = new ResetpwdForm();
4
5          if ($model->load(Yii::$app->request->post())) {
6
7              if($model->resetPassword($id))
8              {
9                  return $this->redirect(['index']);
10             }
11         }
12
13         return $this->render('resetpwd', [
14             'model' => $model,

```

```
15         ]);  
16  
17     }
```

3.3 RBAC授权管理

3.3.1 RBAC设计

RBAC模型（Role-Based Access Control：基于角色的访问控制）。抽象地概括为：Who是否可以对What进行How的访问操作，并对这个逻辑表达式进行判断是否为True的求解过程，也即是将权限问题转换为What、How的问题，Who、What、How构成了访问权限三元组。

角色和权限都实现了树的层次结构，一个角色可能由其他角色或权限构成，权限可以由其他权限构成。

进行权限检查时，一个角色可以指派给一个或多个用户，系统会检查包含该权限的角色是否指派给某个用户，以判断该用户是否包含某些权限。

3.3.2 建立授权数据

1.启用rbac数据库管理组件：

```
1 'authManager' => [  
2     'class' => 'yii\rbac\DbManager',  
3 ]
```

2.创建并补充rbac数据表：

控制台执行数据迁移命令：./yii migrate --migrationPath=@yii/rbac/migrations，在数据库中可以看到新增的数据表；

在console中controller添加rbaccontroller，编辑权限及角色内容后，控制台执行命令：./yii rbac/init，完成对角色及权限的数据库写入。

3.3.3 执行权限检查

在backend/controller各文件中，在需要权限检查的actionxxx函数前添加：

```
1 if(!Yii::$app->user->can('...actionName...')) {  
2     throw new ForbiddenHttpException('对不起，您没有进行该操作的权限')  
3 }
```

3.3.4 添加权限管理页面

1.重写gridview最后一栏中的ActionColumn，添加动作按钮privilege；（与修改密码按钮操作一致，此处省略）

2.添加privilege按钮匿名函数，弹出权限管理界面；（与修改密码按钮操作一致，此处省略）

3.在AdminUserController中添加action函数，该函数执行以下步骤：

①找出所有权限,提供给checkboxlist

```
1 $allPrivileges = AuthItem::find()->select(['name','description'])
2     ->where(['type'=>1])->orderBy('description')->all();
3 foreach ($allPrivileges as $pri)
4     {
5         $allPrivilegesArray[$pri->name]=$pri->description;
6     }
```

②查找当前用户的权限，用于初始化checkboxlist

```
1 $AuthAssignments=AuthAssignment::find()->select(['item_name'])
2     ->where(['user_id'=>$id])->orderBy('item_name')->all();
3
4 $AuthAssignmentsArray = array();
5
6 foreach ($AuthAssignments as $AuthAssignment)
7     {
8         array_push($AuthAssignmentsArray,$AuthAssignment->item_name);
9     }
10
```

③从表单提交的数据,来更新AuthAssignment表,从而用户的角色发生变化（必须勾选其中一个选项，否则无法提交表单）

```
1 if(isset($_POST['newPri']))
2     {
3         AuthAssignment::deleteAll('user_id=:id',[':id'=>$id]);
4
5         $newPri = $_POST['newPri'];
6
7         $arrlength = count($newPri);
8
```



```

9         for($x=0;$x<$arrlength;$x++)
10        {
11            $aPri = new AuthAssignment();
12            $aPri->item_name = $newPri[$x];
13            $aPri->user_id = $id;
14            $aPri->created_at = time();
15
16            $aPri->save();
17        }
18        return $this->redirect(['index']);
19    }
20

```

④使用render渲染checkBoxList表单

```

1 return $this->render('privilege',['id'=>$id,'AuthAssignmentArray'=>$AuthAssignme
2     'allPrivilegesArray'=>$allPrivilegesArray]);

```

4. 前端搭建

使用Yii框架中的Gii模块生成各数据表的Controller、model和CURD的增删查改后，为了实现需求的前端功能，我们对代码进行了如下的修改：

4.1 新闻页面

本项目前端主要分为新闻和博客两方面内容，其中二者代码和功能具有很高的相似度，故本部分只介绍实现新闻功能的代码。

4.1.1 新闻浏览页面

本页面包括新闻的概述、搜索、标签云和最近回复功能，本部分只讲新闻概述功能。

本部分主要实现在frontend/views/news/index.php，关键代码如下：

```

1 <?= ListView::widget([
2     'id'=>'postList',
3     'dataProvider'=>$dataProvider,
4     'itemView'=>'_listitem',//子视图,显示一篇文章的标题等内容.
5     'layout'=>'{items} {pager}',
6     'pager'=>[
7         'maxButtonCount'=>10,
8         'nextPageLabel'=>Yii::t('app','下一页'),
9         'prevPageLabel'=>Yii::t('app','上一页'),

```

```

10     ],
11     ])?>

```

这是通过调用模型的搜索功能对新闻列表进行搜索，且每页最多10篇，否则分页。这之中是通过frontend/controllers/NewsController.php与模型进行连接搜索得到的，代码如下：

```

1  /**
2   * Lists all News models.
3   * @return mixed
4   */
5  public function actionIndex()
6  {
7      $tags=Tag::findTagWeights();
8      $recentNewsComments=Newscomment::findRecentNewsComments();
9
10     $searchModel = new NewsSearch();
11     $dataProvider = $searchModel->search(Yii::$app->request->queryParams);
12
13     return $this->render('index', [
14         'searchModel' => $searchModel,
15         'dataProvider' => $dataProvider,
16         'tags'=>$tags,
17         'recentNewsComments'=>$recentNewsComments,
18     ]);
19 }

```

这部分通过访问common/models/Newssearch.php这一model调用内部的search方法获取新闻。

4.1.2 新闻详细页面

本部分通过frontend/views/news/detail.php实现，代码如下：

```

1  <div class="col-md-9">
2
3      <ol class="breadcrumb">
4          <li><a href="?= Yii::$app->homeUrl;?>">首页</a></li>
5          <li><a href="?= Url::to(['news/index']);?>">新闻列表</a></li>
6          <li class="active"><?= $model->title?></li>
7      </ol>
8
9      <div class="newst">
10         <div class="title">
11             <h2><a href="?= $model->url;?>"><?= Html::encode($model->ti

```

```

12         <div class="author">
13             <span class="glyphicon glyphicon-time" aria-hidden="true"
14             <span class="glyphicon glyphicon-user" aria-hidden="true"
15             </div>
16         </div>
17
18
19     <br>
20
21     <div class="content">
22         <?= HTMLPurifier::process($model->content)?>
23     </div>
24
25     <br>
26
27     <div class="nav">
28         <span class="glyphicon glyphicon-tag" aria-hidden="true"></span>
29         <?= implode(' ', $model->tagLinks);?>
30         <br>
31         <?= Html::a("评论({$model->commentCount})", $model->url. '#comment:
32         最后修改于<?= date('Y-m-d H:i:s', $model->update_time);?>
33     </div>
34 </div>

```

与新闻浏览页面类似，本部分也调用了common/models/Newssearch.php内部的search方法，同时使用了common/models/Newscomment.php统计并展示了新闻评论，代码如下：

```

1  /**
2   * Creates data provider instance with search query applied
3   *
4   * @param array $params
5   *
6   * @return CActiveDataProvider
7   */
8  public function search($params)
9  {
10     $query = Newscomment::find();
11
12     // add conditions that should always apply here
13
14     $dataProvider = new CActiveDataProvider([
15         'query' => $query,
16     ]);
17
18     $this->load($params);

```

```

19
20     if (!$this->validate()) {
21         // uncomment the following line if you do not want to return any rec
22         // $query->where('0=1');
23         return $dataProvider;
24     }
25
26     // grid filtering conditions
27     $query->andFilterWhere([
28         'id' => $this->id,
29         'status' => $this->status,
30         'create_time' => $this->create_time,
31         'userid' => $this->userid,
32         'post_id' => $this->post_id,
33     ]);
34
35     $query->andFilterWhere(['like', 'content', $this->content])
36         ->andFilterWhere(['like', 'email', $this->email])
37         ->andFilterWhere(['like', 'url', $this->url]);
38
39     return $dataProvider;
40 }

```

4.2 搜索功能

这一功能实现了在前端通过标题搜索新闻，代码如下：

```

1 <div class="searchbox">
2     <ul class="list-group">
3         <li class="list-group-item">
4             <span class="glyphicon glyphicon-search" aria-hidden="true"></
5             <?= News::find()->count();?>
6             </li>
7             <li class="list-group-item">
8                 <form action="index.php?r=news/index" id="w1" method="get"
9                 <div class="form-group">
10                     <input type="hidden" name="r" value="news/index">
11                     <input type="text" class="form-control" name="NewsSe
12                     </div>
13                     <button type="submit" class="btn btn-default">搜索</bu
14                 </form>
15             </li>
16         </ul>
17     </div>
18

```

这一代码通过HTML input表单拼接url使其调用common/models/Newssearch.php中按标题搜索的方法对新闻进行搜索。

4.3 标签云功能

这一功能通过总结新闻标签进行展示同时具有搜索功能。

标签云通过在新闻浏览页面调用小部件frontend/components/NewsTagsCloudWidget.php获取，关键代码如下：

```
1 class NewsTagsCloudWidget extends Widget
2 {
3     public $tags;
4
5     public function init()
6     {
7         parent::init();
8     }
9
10    public function run()
11    {
12        $tagString='';
13        $fontStyle=array("6"=>"danger",
14                        "5"=>"info",
15                        "4"=>"warning",
16                        "3"=>"primary",
17                        "2"=>"success",
18        );
19
20        foreach ($this->tags as $tag=>$weight)
21        {
22            $url = Yii::$app->urlManager->createUrl(['news/index','NewsSearch[ta
23            $tagString.='<a href="'. $url.'">'.
24                ' <h'. $weight.' style="display:inline-block;"><span class="l
25                . $fontStyle[$weight].'">'. $tag.'</span></h'. $weight.'"></a>';
26        }
27        sleep(3);
28        return $tagString;
29    }
30 }
```

这一部分调用了common/models/Tag.php的findTagWeights，将获得的tag按权重打印在页面上。这一代码如下：

```
1 public static function findTagWeights($limit=20)
2 {
3     $tag_size_level = 5;
4
5     $models=Tag::find()->orderBy('frequency desc')->limit($limit)->all();
6     $total=Tag::find()->limit($limit)->count();
7
8     $stepper=ceil($total/$tag_size_level);
9
10    $tags=array();
11    $counter=1;
12
13    if($total>0)
14    {
15        foreach ($models as $model)
16        {
17            $weight=ceil($counter/$stepper)+1;
18            $tags[$model->name]=$weight;
19            $counter++;
20        }
21        ksort($tags);
22    }
23    return $tags;
24 }
```

除此之外，它还通过生成包含搜索功能的url调用Newssearch的方法按照tag搜索新闻。

4.4 评论功能

通过调用common/models/Newscomment.php和common/models/NewscommentSearch.php这些评论模型，我们可以实现评论的管理与展示。

4.4.1 发表和浏览评论

本部分通过frontend/views/news/detail.php实现，代码如下：

```
1 <div id="newscomments">
2
3     <?php if($added) {?>
4     <br>
5     <div class="alert alert-warning alert-dismissible" role="alert">
```

```

6         <button type="button" class="close" data-dismiss="alert" aria-label="close"></button>
7
8         <h4>感谢您的回复，我们会尽快审核后发布出来! </h4>
9
10        <p><?= nl2br($commentModel->content);?></p>
11        <span class="glyphicon glyphicon-time" aria-hidden="true"></span>
12        <span class="glyphicon glyphicon-user" aria-hidden="true"></span>
13    </div>
14    <?php }?>
15
16    <?php if($model->commentCount>=1) :?>
17
18    <h5><?= $model->commentCount.'条评论';?></h5>
19    <?= $this->render('_comment',array(
20        'news'=>$model,
21        'comments'=>$model->activeComments,
22    ));?>
23    <?php endif;?>
24
25    <h5>发表评论</h5>
26    <?php
27    $commentModel =new Newscomment();
28    echo $this->render('_guestform',array(
29        'id'=>$model->id,
30        'commentModel'=>$commentModel,
31    ));?>
32
33    </div>
34
35 </div>

```

这部分调用了frontend/controllers/NewsController.php重点actionDetail对评论进行管理，代码如下：

```

1 public function actionDetail($id)
2 {
3     //step1. 准备数据模型
4     $model = $this->findModel($id);
5     $tags=Tag::findTagWeights();
6     $recentNewsComments=Newscomment::findRecentNewsComments();
7
8     $userMe = User::findOne(Yii::$app->user->id);
9     $commentModel = new Newscomment();
10    $commentModel->email = $userMe->email;
11    $commentModel->userid = $userMe->id;

```

```

12
13     //step2. 当评论提交时, 处理评论
14     if($commentModel->load(Yii::$app->request->post()))
15     {
16         $commentModel->status = 1; //新评论默认状态为 pending
17         $commentModel->post_id = $id;
18         if($commentModel->save())
19         {
20             $this->added=1;
21         }
22     }
23
24     //step3. 传数据给视图渲染
25
26     return $this->render('detail',[
27         'model'=>$model,
28         'tags'=>$tags,
29         'recentNewsComments'=>$recentNewsComments,
30         'commentModel'=>$commentModel,
31         'added'=>$this->added,
32     ]);
33 }

```

4.4.2 最新评论功能

这一功能在新闻浏览页面展示最近的新闻评论。

最新评论通过在新闻浏览页面调用小部件frontend/components/RctNewsReplyWidget.php获取，关键代码如下：

```

1 class RctNewsReplyWidget extends Widget
2 {
3     public $recentNewsComments;
4
5     public function init()
6     {
7         parent::init();
8     }
9
10    public function run()
11    {
12        $NewscommentString='';
13
14        foreach ($this->recentNewsComments as $Newscomment)
15        {
16            $NewscommentString.='<div class="news">'.

```



```

17         '<div class="title">'.
18         '<p style="color:#777777;font-style:italic;">'.
19         nl2br($Newscomment->content).'

```

这一代码通过调用了common/models/Newscomment.php的findRecentNewsComments，将获得的最近10条评论打印出来，代码如下：

```

1 public static function findRecentNewsComments($limit=10)
2 {
3     return Newscomment::find()->where(['status'=>2])->orderBy('create_time D
4     ->limit($limit)->all();
5 }

```

5. 版本控制

5.1 项目分工

小组成员：

郭谨、赵一名、林雨豪、陈睿颖

组长：

郭谨 负责前端开发和用户手册编写

组员：

赵一名 负责模板创建、后端文章功能开发和配置文档编写
























林雨豪 负责用户管理、后台评论管理和展示PPT

陈睿颖 负责后端文章功能开发、需求文档和设计文档编写

5.2 Github仓库链接

<https://github.com/nk-guojin/dbis-project>

5.3 项目概览

 nk-guojin vendor			999363b 7 minutes ago	 4 commits
	backend	vendor		7 minutes ago
	common	2.12 update		12 hours ago
	console	2.12 update		12 hours ago
	data	2.13 update		23 minutes ago
	environments	first commit		2 weeks ago
	frontend	vendor		7 minutes ago
	vagrant	first commit		2 weeks ago
	vendor	vendor		7 minutes ago
	.bowerrc	first commit		2 weeks ago
	.gitignore	vendor		7 minutes ago
	LICENSE.md	first commit		2 weeks ago
	README.md	2.12 update		12 hours ago
	Vagrantfile	first commit		2 weeks ago
	codeception.yml	first commit		2 weeks ago
	composer.json	first commit		2 weeks ago
	composer.lock	first commit		2 weeks ago
	docker-compose.yml	first commit		2 weeks ago
	init	first commit		2 weeks ago
	init.bat	first commit		2 weeks ago
	requirements.php	first commit		2 weeks ago
	yii.bat	first commit		2 weeks ago

5.4 参考资料

本项目参考了

<https://www.yiichina.com/video>