

EW_WAR 团队网站实现文档

1、 前台用户页面实现

1.1 网页模板导入

从 <https://www.free-css.com/> 网站中下载对应的网页样式模板，本次使用的模板链接如下：<https://www.free-css.com/free-css-templates/page282/leadmark>，该部分将下载的压缩包解压后放入 frontend\web\assets\frontend 目录下，修改 frontend\view\layouts\main.php 文件，配置 css 和 js 文件，具体需要配置的文件可以在 css 模板的 index.html 中查看，需要注意的是，css 模板中 index.php 的资源配置与本次 Yii 框架网页中需要的配置路径不同，因为我们修改了 css 和 js 文件的位置。

为了 PHP 代码编写，我们需要在模板首页源代码外层加上 `<?php $this->beginPage() ?><?php $this->endPage() ?>`，在 html 的 `<body>` 标签外加上 `<?php $this->beginBody() ?><?php $this->endBody() ?>`。

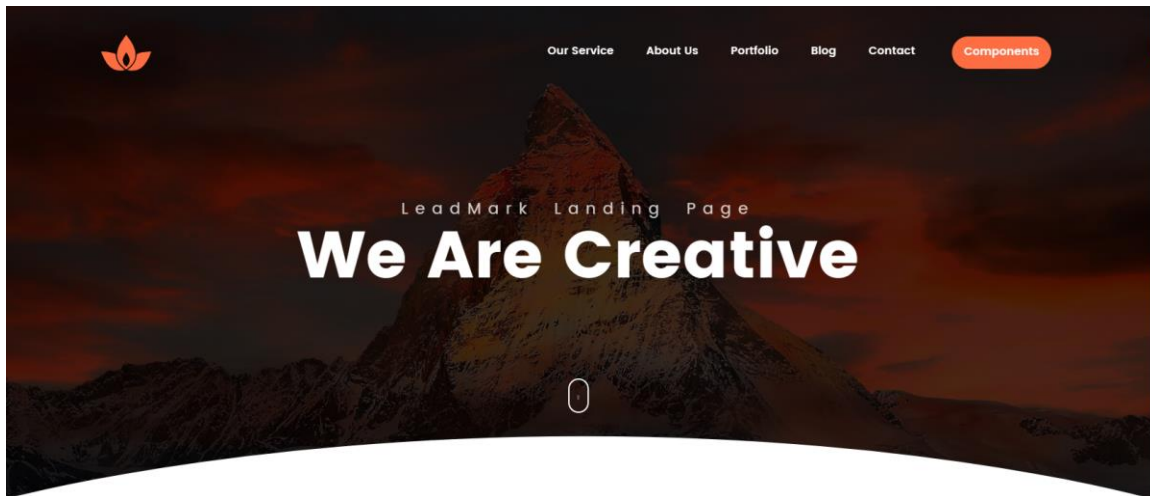
css 配置主要在 `<body>` 标签前的 `<head>` 标签中实现，举例如下：

```
<link rel="stylesheet" href="assets/frontend/leadmark/public_html/assets/css/leadmark.css">
```

js 配置主要在 `<body>` 标签的末尾实现，举例如下：

```
<script src="assets/frontend/leadmark/public_html/assets/js/leadmark.js"></script>
```

模板示例截图如下：



1.2 前台主页

1.2.1 页眉部分

该部分主要在 frontend\view\layouts\main.php 文件中进行编辑，其中，“难民去向”和“最新文章”为页面内跳转，该部分通过 href="#id"和<section>标签中的 id 来实现页内跳转；

```
<!-- main.php 标签列表-->
<li class="nav-item">
    <a class="nav-link" href="#portfolio">难民去向</a>
</li>
<li class="nav-item">
    <a class="nav-link" href="#blog">最新文章</a>
</li>

<!-- index.php 主页 html-->
<section id="portfolio">...</section>
<section id="blog">...</section>
```

其余部分为页面跳转，该部分主要通过\$menuitems[]变量实现，lable 属性用来显示文字，url 属性用来显示相对跳转地址，如"/site/about"表示相对跳转地址为 site 目录下的 about.php 文件。

```
<?php
$menuItems = [
    ['label' => '主页', 'url' => ['/site/index']],
    ['label' => '新闻', 'url' => ['/site/news']],
    ['label' => '关于', 'url' => ['/site/about']],
    ['label' => '援助', 'url' => ['/usupport/index']],
];

if (Yii::$app->user->isGuest) {
    $menuItems[] = ['label' => '注册', 'url' =>
['/site/signup']];
    $menuItems[] = ['label' => '登录', 'url' =>
['/site/login']];
} else {
    $menuItems[] = ['label' => '修改信息', 'url' =>
['/site/modify']];

    $menuItems[] =
"
    <li><span>
        . Html::beginForm(['/site/logout'], 'post')
        . "<button class='btn btn-outline-secondary'>登出("
        . Yii::$app->user->identity->username . ')"
        . "</button>"
        . Html::endForm()
        . '</span></li>';
    }

echo Nav::widget([
    'options' => ['class' => 'nav-item'],
    'items' => $menuItems
]);
?>
```

1.2.2 难民去向部分

采用模板中的案例，修改对应的图片及文字即可。其中，数据在数据库中存储，通过如下方式获取其中的数据：首先在 common/modules 中建立对应的 module 文件，该部分主要文件为 Refugee.php 文件，代码如下：该部分类名与数据库中表名相同，通过 tableName 这一静态函数保证该 module 对应数据库中的数据表

```
/**
 * 难民相关数据，用于在前台显示
 * @property string $nationality    国籍
 * @property string $destination    去向
 * @property string $num            过境人数
 */
class Refugee extends ActiveRecord
{
    public static function tableName()
    {
        return '{{%Refugee}}';
    }
    ...
}
```

在使用时通过如下方式使用：使用 findOne 或者 findAll 函数查找对应记录使用。如下为获取波兰难民数据的代码

```
//查找记录
$Refugee_poland =
\common\models\Refugee::findOne(['destination' => '波兰']);
//获取记录中 num 属性的值
$Refugee_poland->num
```

① 过境人数

该部分通过 num 属性显示，为了显示数据的美观性，将输出数据格式化，方法如下：number_format 函数可以将数字转换为如下格式 9329169->9,329,169

```
<?= number_format($Refugee_poland->num) ?>
```

② ECharts 导入南丁格尔图

该部分主要通过 ECharts 实现，通过 style 中的 height 和 width 调整大小，代码主体采用 ECharts 的案例代码修改获得，数据通过数据库获取，代码如下

```

<div class="col-md-6 col-lg-4 advertising">
<div id="container" style="height: 600%;width: 300%"></div>
<script type="text/javascript"
src="https://fastly.jsdelivrivr.net/npm/echarts@5.4.1/dist/echarts.min.js"></script>
<script type="text/javascript">
    var dom = document.getElementById('container');
    var myChart = echarts.init(dom, null, {
        renderer: 'canvas',
        useDirtyRect: false
    });
    var app = {};

    var option;

    option = {
        legend: {
            top: 'bottom'
        },
        toolbox: {
            show: true,
            feature: {
                mark: { show: true },
                dataView: { show: true, readOnly: false },
                restore: { show: true },
                saveAsImage: { show: true }
            }
        },
        series: [
            {
                name: 'Nightingale Chart',
                type: 'pie',
                radius: [50, 250],
                center: ['50%', '50%'],
                roseType: 'area',
                itemStyle: {
                    borderRadius: 7
                },
                data: [
                    { value: <?= $Refugee_poland->num ?>, name: '波兰

```

③ 视图部分

视图主要获取网页对应图片，通过 href 指定图片来源即可，这里不再赘述。

1.2.3 最新文章部分

1.2.4 页脚部分

主页页脚在 index.php 文件中进行设置，分为“关于本站”、“相关网站”、“发送信息”和“返回首部”几个部分，关于本站部分给出了一部分自己书写的 php 页面和框架中给定的 php 页面采用如下方式指定 `<?=`

`\yii\helpers\Url::to(['/site/index']); ?>`；相关网站则是给定的一些与俄乌战争相关的网站，指定通过 href 指定；发送信息为本站原有代码，通过如下代码接收显示；最后是返回首页，也为页内跳转，与页眉部分类似，这里不再赘述：

```

<section class="banner_main">
  <div class="container-fluid">
    <div class="row d_flex">
      <div class="col-xl-6 col-lg-6 col-md-12">
        <div class="text-bg">
          <span class="caption mb-3 d-block"></span>
          <!--发送通知信息-->
          <?php
            if (isset($message))
              echo "<h1 class=\"heading\">" .
$message . "</h1>";
              if(isset($_GET['message']))
                echo "<h1 class=\"heading\">" .
$_GET['message'] . "</h1>";
              else echo "<h1 class=\"heading\"></h1>";
            ?>
          </div>
        </div>
      </div>
    </div>
  </div>
</section>

```

1.3 新闻页面

新闻页面主要由两部分页面组成：新闻列表页面与新闻内容界面。

(1) 新闻列表界面

新闻列表界面设计如下：


```
<style>
    #news-list {
        list-style-type: none;
        margin: 0 auto;
        padding: 0;
        width: 50%;
        background-color: #f1f1f1;
        border: 1px solid black;
        text-align: center;
    }
    #news-list li a {
        display: block;
        color: #000;
        padding: 8px 16px;
        text-decoration: none;
        font-size: 18px;
        font-weight: bold;
        text-align: center;
        border-bottom: 1px solid black;
    }
    #news-list li a:hover {
        background-color: #555;
        color: white;
    }
</style>
```

(2) 新闻主体界面

新闻主体界面设计如下：

以国前总理斡旋俄乌战争曾获普京承诺不会杀害泽连斯基

VOA
2023-02-06



以色列前总理纳夫塔利·贝内特 (Naftali Bennett) 在周六 (2月4日) 一次公开采访中透露，他在去年俄罗斯入侵乌克兰之后访问莫斯科进行斡旋期间，普京曾承诺他不会杀死乌克兰总统泽连斯基 (Volodymyr Zelenskyy)。贝内特在长达五小时的线上采访中涉及了广泛的议题。

时任以色列总理的贝内特在俄罗斯入侵乌克兰后的头几个星期内，成为一位当时人们不太可能想到的中间调停人，是很少数见到普京的西方国家领导人之一。他去年3月曾突然访问莫斯科，与普京会晤三个小时。

美联社2月5日报道说，尽管贝内特结束战争的斡旋没有取得什么进展，但周六对他的采访暗示了一些当时试图尽快结束战争的幕后外交和其他紧急努力。

贝内特在线上采访中，他在会晤中问及普京他是否意图杀死乌克兰总统泽连斯基。他说，“我问‘你在计划杀死泽连斯基吗？’他说，‘我不会杀死泽连斯基。’我随后对他说，‘我必须要了解你对我承诺，你不会杀死泽连斯基。’他说，‘我不会杀死泽连斯基。’”

贝内特在采访中说，他后来致电泽连斯基，告诉了他普京的承诺。贝内特说，“听着，我稍后会继续，他不会杀死你。”他问，“你确定？”我说，“百分之百 (确定) 他不会杀死你。”

贝内特在采访中还透露，在他斡旋期间，普京放弃了要求解除乌克兰武装的要求，而泽连斯基也承诺不会加入北约组织。

美联社说，目前还没有克里姆林宫的立即回应。俄罗斯此前曾否认乌克兰有关俄罗斯试图暗杀泽连斯基的说法。

报道说，俄罗斯入侵乌克兰之际，贝内特担任以色列总理才六个月，是位仍未受过历练的领导人，但他出人意料地介入了国际调停外交。以色列面对伊朗的威胁认为保持与俄罗斯的友好关系具有战略意义，但同时他也与谋求支持乌克兰的西方国家结盟。

但是，贝内特的和平努力没有取得太大进展，他自己的政治生涯也是非常短暂的。他领导的联合政府去年夏天因为内斗而垮台。

url: <https://www.voachinese.com/a/former-israeli-pm-putin-promised-not-to-kill-zelenskyy-02052023/6948818.html>

请注意，为显示全页面图片对网页进行了缩放，请以实际网页大小为准。

新闻主体界面分为三个部分：标题部分、图片部分、正文部分。

标题部分包含新闻标题、新闻作者（即来源网站，下同）、新闻发布时间三部分。在页面的最上方为 36 号加粗白色居中字体，为新闻标题；新闻标题下起两行，分别为新闻作者、新闻发布时间，均为右对齐 20 号小字。具体实现代码如下：

```
.headerr {  
    font-size: 36px;  
    font-family: Songti;  
    font-weight: bold;  
    text-align: center;  
    color: white;  
}  
  
.authorr {  
    font-size: 20px;  
    font-family: Songti;  
    text-align: right;  
    color: white;  
}  
  
.datee {  
    font-size: 20px;  
    font-family: Songti;  
    text-align: right;  
    color: white;  
}
```

图片部分包含一张与新闻有关的图片，大小为 35%界面宽度，锁定纵横比。具体实现代码如下：

```
.imagee {  
    width: 35%;  
    height: auto;  
    object-fit: contain;  
    margin: 20px auto;  
    display: block;  
}
```

正文部分包含新闻正文和新闻原 url 地址，字体为 15 号左对齐小字。具体实现代码如下：

```
.texttt {  
    font-size: 15px;  
    font-family: "WenQuanYi Zen Hei";  
    line-height: 1.5;  
    text-align: left;  
    color: white;  
    text-indent: 2em;  
}
```

另外，还实现了 container 和 content，用来控制文本显示位置和显示宽度。

在内容获取方面，除新闻正文部分为直接文本，其余均与数据库相连，从数据库中获取内容。与数据库连接，实现内容部分具体实现代码如下：

```
$site1 = \common\models\news::findAll(['id' => 1]);  
if (sizeof($site1) != 0)  
    $site1 = $site1[0];  
else $site1 = null;
```

<body>

<div class="containerr">

<div class="headerr"><?= \$site1->title ?></div>

<div class="authorr"><?= \$site1->author ?></div>

<div class="datee"><?= \$site1->release_date ?></div>

<div class="textt">

<div class="contentt">

<p>以色列前总理纳夫塔利·贝内特（Naftali Bennett）在周六（2月4日）一次公开采访中透露，他在去年俄罗斯入侵乌克兰之后访问莫斯科进行斡旋调和期间，普京曾承诺说他不会杀死乌克兰总统泽连斯基（Volodymyr Zelenskyy）。贝内特在长达五小时的线上采访中涉及了广泛的议题。</p>

<p>时任以色列总理的贝内特在俄罗斯入侵乌克兰后的头几个星期内，成为一位当时人们不太可能想到的中间调解人，是极少数见到普京的西方国家领导人之一。他去年3月曾突然访问莫斯科，与普京会晤三个小时。</p>

<p>美联社2月5日报道说，尽管贝内特结束战争的斡旋没有取得什么进展，但是周六对他的采访揭示了一些当时试图尽快结束战争的背后外交和其他紧急努力。</p>

<p>贝内特在线上采访中说，他在会晤中问及普京他是否意图杀死乌克兰总统泽连斯基。他说，“我问‘你在计划杀死泽连斯基吗？’他说，‘我不会杀死泽连斯基。’我随后对他说，‘我必须要了解你在对我承诺，你不会杀死泽连斯基。’他说，‘我不会杀死泽连斯基。’”</p>

<p>贝内特在采访中说，他后来致电泽连斯基，告诉了他普京的承诺。贝内特说，“‘听着，我刚结束会晤，他不会杀死你。’他问，‘你确定？’我说，‘百分之百（确定）他不会杀死你。’”</p>

<p>贝内特在采访中还透露，在他斡旋期间，普京放弃了谋求解除乌克兰武装的要求，而泽连斯基也承诺不会加入北约组织。</p>

<p>美联社说，目前还没有克里姆林宫的立即回应。俄罗斯此前曾否认乌克兰有关俄罗斯试图暗杀泽连斯基的说法。</p>

<p>报道说，俄罗斯入侵乌克兰之际，贝内特担任以色列总理才六个月，是位仍未受过历练的领导人，但他出人意料地介入了国际调停外交。以色列面对伊朗的威胁认为保持与俄罗斯的良好关系具有战略意义，但同时以色列寻求在乌克兰

1.4 关于页面

该部分主体为团队成员的信息显示，信息通过数据库获取，方式与主页中难民部分的数据获取和使用方式类似，这里不再赘述。为了使页面更美观，调整模板中的 class 参数来调整网页中团队成员所在框的大小，修改后代码如下所示：

```
<div class="col-md-6 my-3 my-md-0">...</div>
```

1.5 援助页面

如“页眉部分节”，在 \$menuItems[] 中添加了 "/usupport/index.php" 后，使用 gii 生成 vi 相应代码（具体步骤见下节）：















效果如下：

Usupports

本页面展示了世界各国对乌克兰的援助,包括不同数量的武器,金钱(美元)等。

Create Usupport

Showing 1-7 of 7 items.

#	Country	Resource	Amount	Time	
1	America	Armored vehicle	350	2022-02-26	 
2	America	Infantry fighting vehicle	140	2022-03-31	 
3	England	Monster Hummer	200	2022-04-04	 
4	England	Self-propelled gun	30	2022-04-14	 
5	England	Tank	14	2022-04-16	 
6	Germany	Infantry fighting vehicle	40	2022-05-11	 
7	Germany	money	1100000	2022-02-24	 

对 view/Model 名/index.php 做出对应修改，可使跳转后页面显示不同文字、图片等，

```
15  ?>
16  <div class="usupport-index">
17
18      <h1><?= Html::encode($this->title) ?></h1>
19      <p>
20          <font size="6">本页面展示了世界各国对乌克兰的援助,包括不同数量的武器,金钱(美元)等。</font>
21      </p>
22      <p>
```

1.6 使用 gii 生成代码步骤

1.6.1 准备工作 Model Generator

- ①首先确保 MySQL 服务正常，且网页能连接数据库（通过 common/config/main-local.php 设置）。
- ②以...frontend/web 作为服务器根目录打开后，在网页输入?r=gii，点击 model generator 进入如下页面，在 table name 中选择数据库表名，使用该表名的首字母大写形式作为 Model Class Name。
- ③尤其注意下方的 namespace 选择，确定将 model 文件生成在 frontend 或 common 或 backend 对应的 model 文件夹下。（处于实现尽量简便的考量，将模型类生成在了 frontend 中，其实生成在 common 文件夹中可实现前后台共用）

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.

Database Connection ID

db

☐ Use Table Prefix

☒ Use Schema Name

Table Name

collision

☐ Standardize Capitals

☐ Singularize

Model Class Name

Collision

Namespace

frontend\models

1.6.2Generate Model

其他选项保持默认不变即可，注意到所有的模型类都继承自 yii\db\ActiveRecord 类，ActiveRecord 类可以说是 yii 框架的精华核心所在。
点击 preview，看到生成代码如下：

```

class Collision extends \yii\db\ActiveRecord
{
    /**
     * {@inheritdoc}
     */
    public static function tableName()
    {
        return 'collision';
    }

    /**
     * {@inheritdoc}
     */
    public function rules()
    {
        return [
            [['place', 'time'], 'required'],
            [['time'], 'safe'],
            [['rforce', 'uforce', 'rinjury', 'uinjury'],
'integer'],
            [['place'], 'string', 'max' => 255],
            [['place', 'time'], 'unique', 'targetAttribute' =>
['place', 'time']],
        ];
    }

    /**
     * {@inheritdoc}
     */
    public function attributeLabels()
    {
        return [
            'place' => 'Place',
            'time' => 'Time',
            'rforce' => 'Rforce',
            'uforce' => 'Uforce',
            'rinjury' => 'Rinjury',
            'uinjury' => 'Uinjury',
        ];
    }
}

```


可以看到自动生成的代码较为简明规范，定义了表属性及相应结构，表具体的属性也在文件前以注释形式给出，为我们自己写代码提供了很好的参考。本文中战争相关的 model 均采用类似结构。

确认是否需要 overwrite 覆盖原有文件后，点击 generate

PreviewGenerate

Click on the above **Generate** button to generate the files selected below:

☒ Create ☒ Unchanged ☒ Overwrite

Code File	Action	
models\Collision.php diff	overwrite	<input type="checkbox"/>

1.6.3CRUD Generator

CRUD 即“增删改查”的首字母缩写，主要实现数据库的数据操作相关功能，并且也生成 view 类的代码，填写之前生成 model 类的位置，以及想要生成的 search model 类和 controller 类的位置（重要）。

CRUD Generator

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

Model Class

frontend\models\Uwardamage

Search Model Class

frontend\models\uwardamageSearch

Controller Class

frontend\controllers\UwardamageController

1.6.4generate controlller and view

同样确认无误后点击 preview->generate, 生成的 controller 类大体代码如下：

```

<?php
namespace frontend\controllers;
use frontend\models\collision;
use frontend\models\collisionSearch;
use yii\web\Controller;
use yii\web\NotFoundHttpException;
use yii\filters\VerbFilter;
/**
 * CollisionController implements the CRUD actions for
 * Collision model.
 */
class CollisionController extends Controller
{
    /**
     * @inheritdoc
     */
    public function behaviors()
    {
        return array_merge(
            parent::behaviors(),
            [
                'verbs' => [
                    'class' => VerbFilter::className(),
                    'actions' => [
                        'delete' => ['POST'],
                    ],
                ],
            ],
        );
    }

    /**
     * Lists all Collision models.
     *
     * @return string
     */
    public function actionIndex()
    {
        $searchModel = new CollisionSearch();
        $dataProvider =

```

可以看到生成 controller 类需要先引入对应的 model 类和 modelSearch 类。

CRUD Generator 同时生成了 view 类，其中包含了每个操作模型的方法单独成文件（create、update 等），update 文件内容如下，负责在页面接受数据传递给模型：

```
$this->title = 'Update Usupport: ' . $model->country;
$this->params['breadcrumbs'][] = ['label' => 'Usupports',
    'url' => ['index']];
$this->params['breadcrumbs'][] = ['label' => $model->country,
    'url' => ['view', 'country' => $model->country, 'resource' =>
    $model->resource, 'time' => $model->time]];
$this->params['breadcrumbs'][] = 'Update';
?>
<div class="usupport-update">

    <h1><?= Html::encode($this->title) ?></h1>

    <?= $this->render('_form', [
        'model' => $model,
    ]) ?>

</div>
```

注意到视图 view 中每一个文件，在 controller 类里也有相应的 action 某某（actionCreate、actionUpdate 等），controller 类是连接 view 和 model 类的桥梁。

在 GridView 小部件是从数据提供者获取数据，并以一个表格的形式呈现数据。表中的每一行代表一个单独的数据项，列表示该项目的属性。可以使用 yii\grid\SerialColumn, yii\grid\CheckboxColumn 和 yii\grid\SerialColumn 进行定制样式。

1.7 最新文章部分

修改文章主页，基本都在\frontend\views\site\index.php
主要更改如下：

```
  
    <div class="card-body">  
        <h6 class="card-title">俄乌历次冲突汇总  
</h6>  
        <p>整理了俄乌双方各次冲突的时间、地点、投入兵  
力、伤亡情况等</p>  
        <a  
href="index.php?r=collision%2Findex" class="small text-muted">  
跳转到数据界面</a>
```

插入自己设定的图片，跳转的超链接为 gj 生成的 view 类下的 index.php 界面。之后跳转到的数据库页面实现思路与“援助”部分大体相同，不再赘述。

1.8 注册和登录

首先，需要在 common\models 中建立 User.php 来维护用户类，其代码如下(只展示核心代码)：

```

class User extends ActiveRecord implements IdentityInterface
{
    /*
     * @return 返回的是数据库中存储用户使用的表名
     */
    public static function tableName()
    {
        return '{{%User}}';
    }
    /*
     * 通过身份证号寻找唯一的用户
     */
    public static function findIdentity($account)
    {
        return static::findOne(['account' => $account]);
    }
    // 其他函数，如 getter 和 setter，这里省略
}

```

1.8.1 注册

需要在\frontend\models 中建立 SignupForm 的模型，用来维护注册信息表单，SignupForm 类代码如下：

首先，声明注册表单需要的四个信息变量，即账号、密码、用户名以及电话号码

```
class SignupForm extends Model
{
    public $account;
    public $password;
    public $username;
    public $tel;
    //...
}
```

接着，通过 `rules` 函数设置每个输入信息文本的规则，例如身份证号账号必须为 18 位且不能为空，用户名要在 1-10 位之间且非空等等，代码如下所示：

```
public function actionSignup()
{
    $model = new SignupForm();
    if ($model->load(Yii::$app->request->post()) &&
$model->signup()) {
        Yii::$app->session->setFlash('success', 'Thank you
for registration. Please check your inbox for verification
email.');
```

```
        return $this->goHome();
    }
    return $this->render('signup', [
        'model' => $model,
    ]);
}
```

即，先新建一个注册表单类，如果注册成功，则返回首页；否则就渲染一个名为 'signup' 的视图，并将该视图的 model 设置为当前的 model，因此，我们需要在视图模块中(\frontend\views)创建 signup.php 作为注册页面的视图，下面是主题部分的代码：

```

<div class="row justify-content-center">
    <div class="col-lg-5">
        <?php $form = ActiveForm::begin(['id' => 'form-
signup']); ?>
        <?= $form->field($model,
'account')->textInput(['autofocus' => true]) ?>
        <?= $form->field($model,
'username')->passwordInput() ?>
        <?= $form->field($model,
'tel')->passwordInput() ?>
        <?= $form->field($model,
'password')->passwordInput() ?>
        <center>
            <div class="form-group">
                <?= Html::submitButton('Signup', ['class'
=> 'btn btn-primary', 'name' => 'signup-button']) ?>
            </div>
        </center>
        <?php ActiveForm::end(); ?>
    </div>
</div>

```

可以看到，这里通过 ActiveForm 的类方法建立文本输入框和提交按钮，文本输入框通过调用 field 方法，对模型 model(也就是前面 SiteController 中新建立的 SignupForm)的相应属性进行填写。其中 textInput 是普通的可见的输入，passwordInput 是将输入内容设置为不可见的形式。提交按钮通过 Html::submitButton 进行创建即可。

1.8.2 登陆

登陆的实现流程和注册基本一致，除了以下的几点不同：首先是登陆时只需要输入账号和密码，因此对应的 LoginForm 类只需要设置账号和密码作为其成员变量即可。其 SiteController 中的 action 函数代码如下：


```

public function actionLogin()
{
    if (!Yii::$app->user->isGuest) {
        return $this->goHome();
    }
    $model = new LoginForm();
    if ($model->load(Yii::$app->request->post()) &&
        $model->login()) {
        return $this->goBack();
    }
    $model->password = '';
    return $this->render('login', [
        'model' => $model,
    ]);
}

```

即，如果已经登录，就返回主页。否则新建一个注册表单类，如果注册成功则回到主页，否则停留在所渲染的 login 视图页面，此时密码默认为空。

1.9 修改用户信息

与注册和登陆类似，也需要在 models 中建立 ModifyForm.php 用于维护修改信息的表单类，其中的成员变量包括新的密码、新的用户名等要修改的信息。在 SiteController 中实现如下 action 方法：

```

public function actionModify()
{
    if (Yii::$app->user->isGuest) {
        return $this->goHome();
    }
    $model = new ModifyForm();
    if ($model->load(Yii::$app->request->post())) {
        if ($model->setMyUser()) {
            $model->setInfo();
            Yii::$app->user->logout();
            return $this->redirect(array('/site/index',
                'message' => "信息修改成功，请重新登录。"
            ));
        } else {
            return $this->render('modify', [
                'model' => $model,
                'message' => "用户名或密码错误"
            ]);
        }
    }
    return $this->render('modify', [
        'model' => $model,
    ]);
}

```

即，如果提交了表单且密码正确，则修改信息并登出当前和账号，否则留在当前页面。其中修改信息的视图与注册和登陆页面的布局类似，不加赘述

2.0 评论页面

对于每一则新闻的评论均单独处理，以 id 为 1 的新闻为例，在 SiteController 中首先通过 `findAll(['id'] => 1)` 查找 id 为 1 的新闻，再通过 `findAll(['New_id'])` 查找所有 New_id 为 1 的评论，然后新建一个 CommentForm 类，该类在 models 中进行实现。随后，如果发布评论按钮提交成功，则仍重定向路由到当前页面：

```

        if ($model->submit())
        {
            return $this->redirect(array('/site/comment1', 'message'
=> '发布成功!', 'id' => 1));
        }
    }

```

同时，在视图模块建立对应的评论界面，其中发布评论的文本框和提交按钮与注册登录基本一致。主要介绍展示评论的实现代码：

```

<?php if ($comment1 != null)
    foreach ($comment1 as $_comment) : ?>
        <div class="comments-list-wrap">
            <div class="comment-list">
                <div class="single-comment-body">
                    <div class="comment-text-body">
                        <h4> <?=
$_comment->author ?></h4>
                        <p><?= $_comment->content ?></p>
                    </div>
                </div>
            </div>
        </div>
    </div>
<?php endforeach; ?>

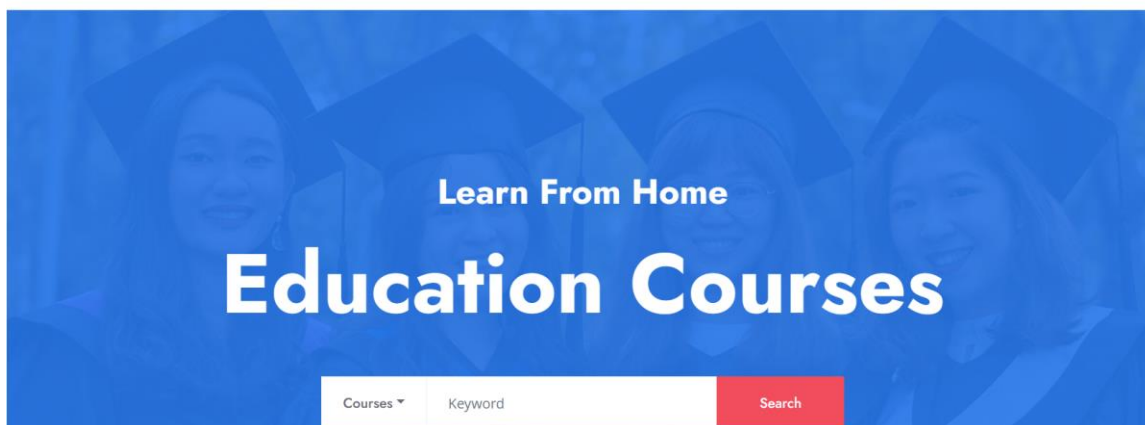
```

即，通过 php 的 foreach 对数据表进行循环遍历即可，这里注意需要在数据表非空的条件下进行

2、 后台管理页面实现

2.1 网页模板导入

该部分与前台的模板导入类似，这里只给出模板网址和样式图：<https://www.free-css.com/free-css-templates/page282/edukate>



2.2 用户信息管理

在控制器中加入 action，即点击相应按钮后渲染 userinfo.php 页面，为了防止非管理员用户访问，在 action 开始时通过 if 分支进行限制，代码如下：

```
public function actionUserinfo()  
{  
    if (Yii::$app->user->identity->type==0)  
    {  
        return $this->goHome();  
    }  
    return $this->render('userinfo');  
}
```

userinfo.php 页面主要获取所有非管理员用户的信息并统计数量，较为容易，这里不再赘述。

2.3 评论内容管理

首先，在后台控制器中实现以下的 action 方法：

```
$news = \common\models\news::find()->all();  
$comment = \common\models\Comment::find()->all();  
$cnt = sizeof($comment);
```

然后双重循环遍历评论表：

```
foreach ($news as $_news) : ?>  
    <?php $comment_temp =  
    \common\models\Comment::findAll(['New_id' => $_news->id]);?>  
    <?php if ($comment_temp != null)  
        foreach ($comment_temp as $_comment) : ?>  
            <div class="d-flex justify-content-between border-  
bottom px-4">  
                <h6 class="text-white my-3"><?=  
$_comment->New_id?></h6>  
                <h6 class="text-white my-3"><?=  
$_comment->id?></h6>  
                <h6 class="text-white my-3"><?=  
$_comment->content?></h6>  
                <h6 class="text-white my-3"><?=  
$_comment->author?></h6>  
            </div>  
        <?php endforeach; ?>  
    <?php endforeach; ?>
```

最后设置删除评论的输入文本框和按钮：

```
<?php $form = ActiveForm::begin(); ?>
    <div class="form-group">
        <h2><?= $form->field($model,
'id')->textInput(['autofocus' => true]) ?></h2>
    </div>
    <div class="form-group">
        <?= Html::submitButton('提交', ['class' => 'btn btn-
primary']) ?>
    </div>
<?php ActiveForm::end(); ?>
```