

实现文档

## 一、分工情况

frontend:

前台展示页面 php 逻辑实现：詹洪骁

前台页面设计与 html 实现：张桐鹤

backend:

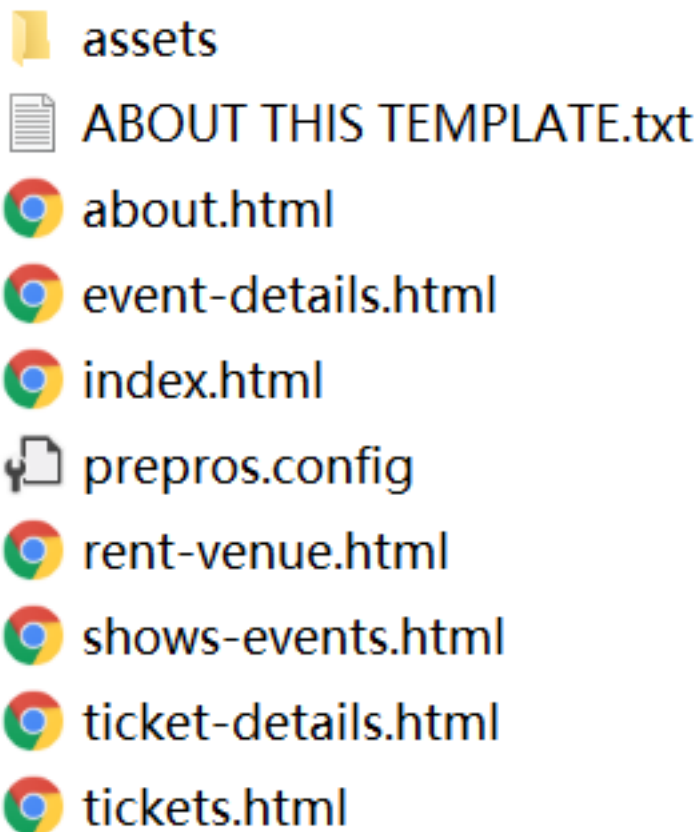
后端界面设计与实现：廖湔睿

数据库设计与爬虫：马世元

## 二、实现过程

### 1.前台

首先我们选取了一个前台模板，是一个打包好的文件，包括了主要的 html 网页和 assets 文件夹包括了我们需要的资源文件。

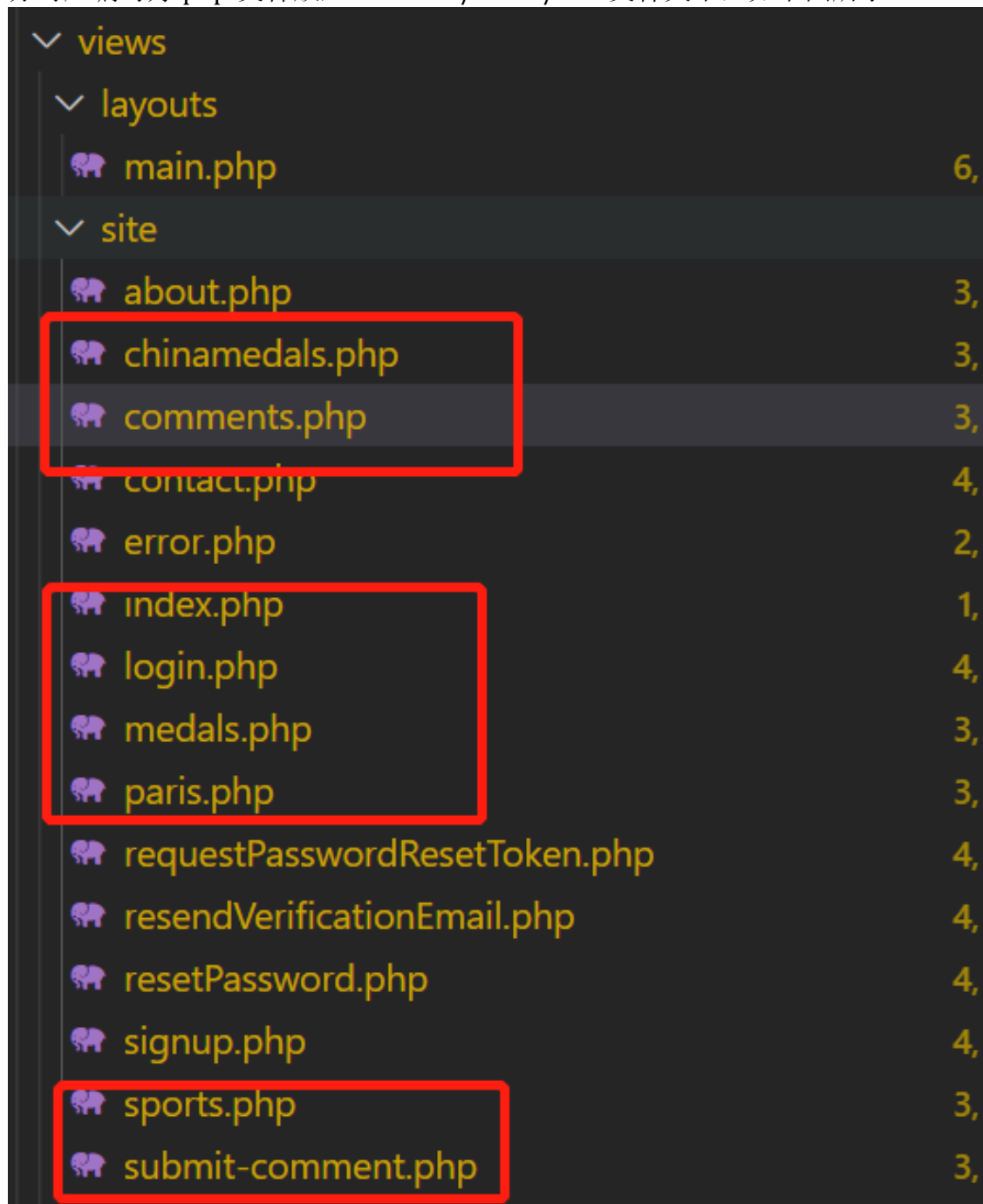


之后我们对模板进行了设计与重构，设计之后的主页如下图所示：



之后是将前台模板整合进 yii2 框架的过程，yii2 框架中的页面都是 php 文件，主页相关的 php 文件应放在“frontend/views/site”文件夹下，且其中的框架是单独写在 layout/main.php 中。而我们的网页模板是 html 文件，其中 header 和 footer 是框架部分，于是我们将框架部分放入 layout 中，再将每一个页面的 content 部

分对应编写好 php 文件放入 frontend/views/site 文件夹中，如下图所示：



完成此处的 php 文件编写后，要在 frontend/controllers/SiteController 中，编写他们对应的 action 方法，方法中要 render 渲染我们的页面，我们才能通过输入参数访问到这些页面。

```

/**
 * Displays medals page.
 *
 * @return mixed
 */
public function actionMedals()
{
    return $this->render('medals');
}

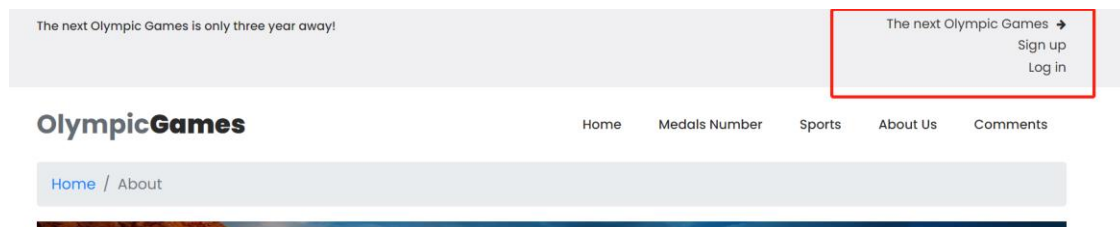
```

例如，写完上面的 `actionMedals` 方法，就可以用 <http://localhost/advanced/frontend/web/index.php?r=site/medals> 访问到我们的奖牌榜页面。

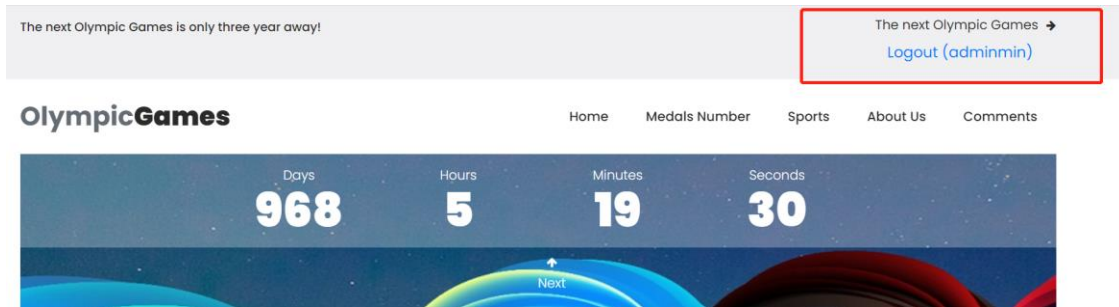
其中要注意的是，`action` 方法的命名一定要保证 `action` 作前缀，后面的方法每个单词的首字母要大写，这样才能对应到 `views` 中的网页文件，对其进行操作。

之后按照地址规范，修改原来 `html` 网页中各个跳转到超链接，改成 `yii2` 框架中对应的路径即可。并且要把资源文件夹 `asset`，放入 `frontend/web/` 下。

在这里需要说明，因为前台利用 `yii2` 模板实现了注册、登录登出的功能，设计了按钮样式，想要登陆后端就需要从前台进行注册，前台的登录也实现了但在前台并没有实际的功能需要登陆才能使用。登录的设计如下所示：



登出的按钮是这样的：



下面是主要展示奥运会奖牌榜的实现过程：

#####

首先我们进行了数据的爬取，并且构建数据库。

爬虫的主体为三部分：1. 获取目标网页的 html 文档 2. 获取 html 里面我们想要的  
数据 3. 将数据写入 excel 中，最后把 excel 导入到 mysql 中即可。

#### 1. 获取目标网页的 html 文档

爬虫的思路就是模拟浏览器访问，为此我们要建立一个头结构 header，用以存放请求的相关参数，这些参数可以直接从浏览器的 f12 里获得，点击第一个数据包请求里面的报头即可拿来使用。



这里我们不需要全部数据，只需要 accept, accept-encoding, accept-language, connection, user-agent 五部分数据即可。

构建好请求头后就可以模拟访问了，使用 requests.get()方法。

```

header = {
    'Accept': 'text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp',
    'Accept-Encoding': 'gzip, deflate',
    'Accept-Language': 'zh-CN,zh;q=0.9',
    'Connection': 'keep-alive',
    'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4431.24 Safari/537.36'
}
timeout = random.choice(range(80, 180))
while True:
    try:
        rep = requests.get(url, headers = header, timeout = timeout)
        rep.encoding = 'utf-8'
        break
    except:
        continue

```

timeout 是响应时间，为一个随机数，目的是防止被浏览器认定为爬虫拒绝访问。


## 2. 从 html 中找到我们想要的数据

上一步会返回 html 文档，这里我们首先要调用解析器对其进行解析。构造 BeautifulSoup 类对象

```
bs = BeautifulSoup(html_text, "html.parser")
```

其中“html.parser”是最通用的浏览器解析器。

解析完毕后要研究 html 文档的结构，打开浏览器进入开发者模式，找到我们想要数据的位置，这里我们以 2020 年奥运会奖牌榜为例，



```

::before
<div class="aoyunhui_19621_zjp_ind01">
  <div class="info_title" data-spm-anchor-id="0.Pe9yiMu0gzhd.ES3pdHUFxT0w.11">...</div>
  <div class="time" id="times">...</div>
  <div class="box">
    <table cellspacing="0" cellpadding="0">
      <thead>...</thead>
      <tbody id="medal_list1">
        <tr class="white" style="display: table-row;">...</tr>
        <tr class="gray" style="display: table-row;">
          <td>2</td>
          <td class="country">
            <a href="//2020.cctv.com/medal_list/details/index.shtml?spm=0.Pe9yiMu0gzhd.ES3pdHUFxT0w.6&countryId=CHN" target="_blank" data-spm-anchor-id="0.Pe9yiMu0gzhd.ES3pdHUFxT0w.6">...</a> == $0
          </td>
          <td>...</td>
          <td>...</td>
          <td>...</td>
        </tr>
        <tr class="white" style="display: table-row;">...</tr>
        <tr class="gray" style="display: table-row;">...</tr>
        <tr class="white" style="display: table-row;">...</tr>
        <tr class="gray" style="display: table-row;">...</tr>
        <tr class="white" style="display: table-row;">...</tr>
        <tr class="gray" style="display: table-row;">...</tr>
        <tr class="white" style="display: table-row;">...</tr>
        <tr class="gray" style="display: table-row;">...</tr>
        <tr class="white" style="display: table-row;">...</tr>
        <tr class="gray" style="display: table-row;">...</tr>
      </tbody>
    </table>
  </div>
</div>

```

“中国”这一行在源码中的位置为：

body->div(class : box)->table->tbody->tr

我们还可以看到 tbody 下还有很多个 tr 子项，事实上每一个 tr 对应一行国家数据。所以我们要做的事就是一层层的进入到 html 的结构中，最终找到这些 tr 并提取出我们要的元素。

```

def get_data(html_text):
    data = []
    bs = BeautifulSoup(html_text, "html.parser")
    body = bs.body
    div_father = body.find('div', {'class': 'aoyunhui_19621_zjp_ind01'})
    div = div_father.find('div', {'class': 'box'})
    table = div.find('table')
    tbody = table.find('tbody')
    items = tbody.find_all('tr', {'class': 'white'})
    for item in items:
        temp = []
        tds = item.find_all('td')
        num = tds[0].string
        icon = tds[1].find('a').find('img').find('img')['src']
        country = tds[2].find('a').string
        gold = tds[3].find('a').string
        silver = tds[4].find('a').string
        cooper = tds[5].find('a').string
        total = tds[6].find('a').string
        temp.append(num)
        temp.append(icon)
        temp.append(country)
        temp.append(gold)
        temp.append(silver)
        temp.append(cooper)
        temp.append(total)
        data.append(temp)

    return data

```

3. 将数据写入到 excel 中  
这一步就简单了，只需调用 csv 库的 writer 函数即可写出为.csv 表格文件

```

def write_data(data, name):
    file_name = name
    with open(file_name, 'a', errors='ignore', newline='') as f:
        f_csv = csv.writer(f)
        f_csv.writerows(data)

```

爬虫结果展示：

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
排名		金牌	银牌	铜牌	总数	icon									
1	美国	39	41	33	113	<a href="http://p1.img.cctvpic.com/sports/data/olympic/teamimg/USA.png">http://p1.img.cctvpic.com/sports/data/olympic/teamimg/USA.png</a>									
2	中国	38	32	18	88	<a href="http://p1.img.cctvpic.com/sports/data/olympic/teamimg/CHN.png">http://p1.img.cctvpic.com/sports/data/olympic/teamimg/CHN.png</a>									
3	日本	27	14	17	58	<a href="http://p1.img.cctvpic.com/sports/data/olympic/teamimg/JPN.png">http://p1.img.cctvpic.com/sports/data/olympic/teamimg/JPN.png</a>									
4	英国	22	21	22	65	<a href="http://p1.img.cctvpic.com/sports/data/olympic/teamimg/GBR.png">http://p1.img.cctvpic.com/sports/data/olympic/teamimg/GBR.png</a>									
5	ROC	20	28	23	71	<a href="http://p1.img.cctvpic.com/sports/data/olympic/teamimg/ROC.png">http://p1.img.cctvpic.com/sports/data/olympic/teamimg/ROC.png</a>									
6	澳大利亚	17	7	22	46	<a href="http://p1.img.cctvpic.com/sports/data/olympic/teamimg/AUS.png">http://p1.img.cctvpic.com/sports/data/olympic/teamimg/AUS.png</a>									
7	意大利	10	10	20	40	<a href="http://p1.img.cctvpic.com/sports/data/olympic/teamimg/ITA.png">http://p1.img.cctvpic.com/sports/data/olympic/teamimg/ITA.png</a>									
8	德国	10	11	16	37	<a href="http://p1.img.cctvpic.com/sports/data/olympic/teamimg/GER.png">http://p1.img.cctvpic.com/sports/data/olympic/teamimg/GER.png</a>									
9	法国	10	12	11	33	<a href="http://p1.img.cctvpic.com/sports/data/olympic/teamimg/FRA.png">http://p1.img.cctvpic.com/sports/data/olympic/teamimg/FRA.png</a>									
10	荷兰	10	12	14	36	<a href="http://p1.img.cctvpic.com/sports/data/olympic/teamimg/NED.png">http://p1.img.cctvpic.com/sports/data/olympic/teamimg/NED.png</a>									
11	古巴	7	3	5	15	<a href="http://p1.img.cctvpic.com/sports/data/olympic/teamimg/CUB.png">http://p1.img.cctvpic.com/sports/data/olympic/teamimg/CUB.png</a>									
12	新西兰	7	6	7	20	<a href="http://p1.img.cctvpic.com/sports/data/olympic/teamimg/NZL.png">http://p1.img.cctvpic.com/sports/data/olympic/teamimg/NZL.png</a>									
13	巴西	7	6	8	21	<a href="http://p1.img.cctvpic.com/sports/data/olympic/teamimg/BRA.png">http://p1.img.cctvpic.com/sports/data/olympic/teamimg/BRA.png</a>									
14	加拿大	7	6	11	24	<a href="http://p1.img.cctvpic.com/sports/data/olympic/teamimg/CAN.png">http://p1.img.cctvpic.com/sports/data/olympic/teamimg/CAN.png</a>									
15	韩国	6	4	10	20	<a href="http://p1.img.cctvpic.com/sports/data/olympic/teamimg/KOR.png">http://p1.img.cctvpic.com/sports/data/olympic/teamimg/KOR.png</a>									
16	匈牙利	6	7	7	20	<a href="http://p1.img.cctvpic.com/sports/data/olympic/teamimg/HUN.png">http://p1.img.cctvpic.com/sports/data/olympic/teamimg/HUN.png</a>									
17	牙买加	4	1	4	9	<a href="http://p1.img.cctvpic.com/sports/data/olympic/teamimg/JAM.png">http://p1.img.cctvpic.com/sports/data/olympic/teamimg/JAM.png</a>									
18	挪威	4	2	2	8	<a href="http://p1.img.cctvpic.com/sports/data/olympic/teamimg/NOR.png">http://p1.img.cctvpic.com/sports/data/olympic/teamimg/NOR.png</a>									
19	肯尼亚	4	4	2	10	<a href="http://p1.img.cctvpic.com/sports/data/olympic/teamimg/KEN.png">http://p1.img.cctvpic.com/sports/data/olympic/teamimg/KEN.png</a>									
20	捷克	4	4	3	11	<a href="http://p1.img.cctvpic.com/sports/data/olympic/teamimg/CZE.png">http://p1.img.cctvpic.com/sports/data/olympic/teamimg/CZE.png</a>									
21	波兰	4	5	5	14	<a href="http://p1.img.cctvpic.com/sports/data/olympic/teamimg/POL.png">http://p1.img.cctvpic.com/sports/data/olympic/teamimg/POL.png</a>									
22	乌兹别克 斯坦	3		2	5	<a href="http://p1.img.cctvpic.com/sports/data/olympic/teamimg/UZB.png">http://p1.img.cctvpic.com/sports/data/olympic/teamimg/UZB.png</a>									
23	斯洛文尼 亚	3	1	1	5	<a href="http://p1.img.cctvpic.com/sports/data/olympic/teamimg/INA.png">http://p1.img.cctvpic.com/sports/data/olympic/teamimg/INA.png</a>									
24	保加利亚	3	1	2	6	<a href="http://p1.img.cctvpic.com/sports/data/olympic/teamimg/DOM.png">http://p1.img.cctvpic.com/sports/data/olympic/teamimg/DOM.png</a>									
25	比利时	3	1	3	7	<a href="http://p1.img.cctvpic.com/sports/data/olympic/teamimg/BEL.png">http://p1.img.cctvpic.com/sports/data/olympic/teamimg/BEL.png</a>									
26	塞尔维亚	3	1	5	9	<a href="http://p1.img.cctvpic.com/sports/data/olympic/teamimg/SRB.png">http://p1.img.cctvpic.com/sports/data/olympic/teamimg/SRB.png</a>									

虽然不同网站的结构不同，爬虫程序也应作相应的调整，但是对于本项目的奥运网址来说，网站结构大同小异，且爬虫程序省去了格式化的过程，处理起来十分方便。

然后根据设计好的数据库的表结构，用 `gii` 创建对应的 `model` 和 `curd` 代码，如下图所示：



# Model Generator

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

Database Connection ID

db

db

☐ Use Table Prefix

☒ Use Schema Name

Table Name

world\_medal\_show

☐ Standardize Capitals

☐ Singularize

Model Class Name

WorldMedalShow

Namespace

frontend\models

CURD 生成的时候，要注意几个地方，都是踩过的坑：

Model Generator	>
<b>CRUD Generator</b>	>
Controller Generator	>
Form Generator	>
Module Generator	>
Extension Generator	>

## CRUD Generator

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

**Model Class**

**Search Model Class**

**Controller Class**

**View Path**

**Base Controller Class**

yii\web\Controller

**Widget Used in Index Page**

GridView

首先最开始的 **Model Class** 必须填写前一步生成的 **Model** 名，且这个表必须要有主键。

下面的 **Search Model Class** 主要是生成对该数据表的查询模块。

**Controller Class** 是负责对这个数据表进行的各种操作，包括增删改查等等，这里命名一定是首字母大写，**Controller** 的 C 要大写，其余单词的首字母都需要小写，不过我们只对奥运会奖牌信息进行展示，并没有修改的需求。

**View Path** 要注意对应的文件夹的名字一定要全部小写，否则 **render** 方法渲染后会找不到该网页，报错 **404 Not Found**，不过实际上我们实现的过程并不需要生成 **View** 模块，我们单独自己设计视图的样式。

上面是 **gii** 的过程，通过 **gii** 我们可以得到两个 **Model**，一个负责和数据库中的表建立联系，另一个 **Search** 模型负责从数据库中读取信息，也可以做搜索操作。我们还得到一个 **Controller** 文件，和这一部分用不到的视图文件（如果使用 **gii** 自带的视图，就需要生成这部分）。

之后我们利用生成好的 **Model**，来进行数据库信息的搜索和展示，主体代码如下：

```
EOT;  
echo $html;}  
?>
```

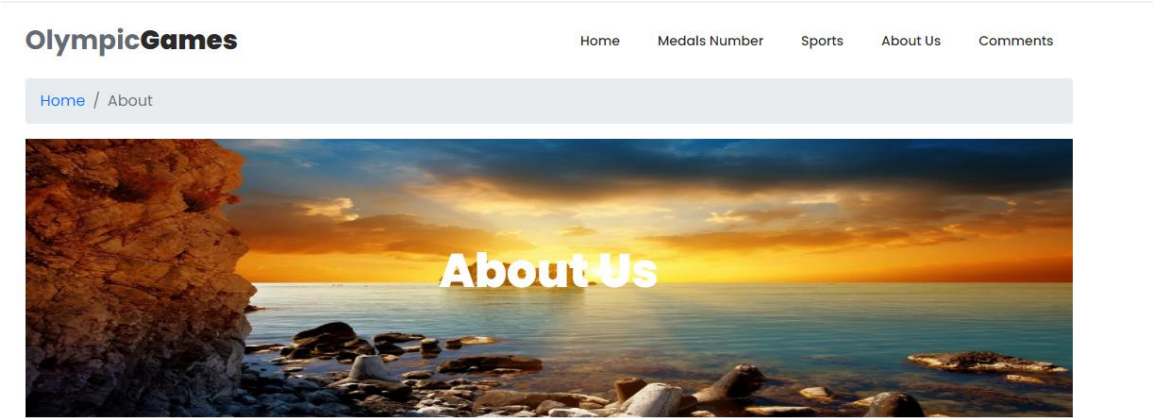
具体显示效果如下：



还有一些其他页面的设计，比如 **About** 页面提供了我们的个人信息和个人作业网址，**Sports** 页面可以查看一部分奥运会项目的运动的信息和介绍，**Comments** 页面提供了留言功能。

Comments 页面：

About 页面：



卧龙凤雏捏队

Member1

张桐鹤 1911518

Member2

Sports 页面：



[FirstPage](#) [NextPage](#)

[Find More](#)

## Swimming

[Discover More](#)



Swimming is the self-propulsion of a person through water, usually for recreation, sport, exercise, or survival.

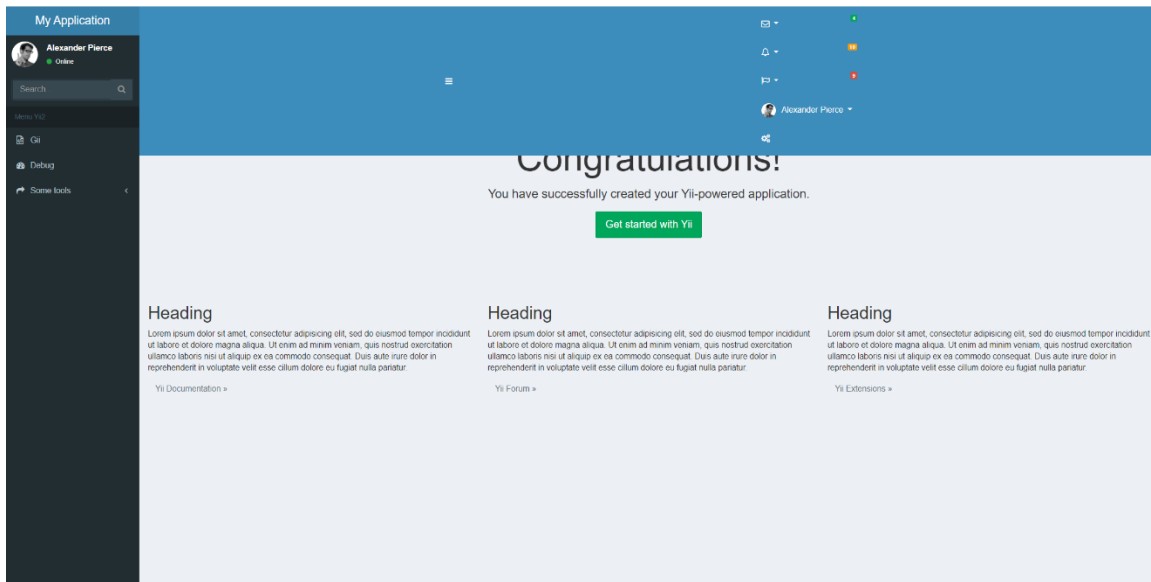
## 2.后台

后台部分的第一步工作仍然是选择一个模板。这里我从 GitHub 上找到了一个可以直接适配 yii2 的模板，网址为：<https://github.com/dmstr/yii2-adminlte-asset>

我们可以使用 `composer` 指令一键安装，在 `\advanced` 目录下进入 `cmd`，输入：  
`composer require dmstr/yii2-adminlte-asset "^2.1"`

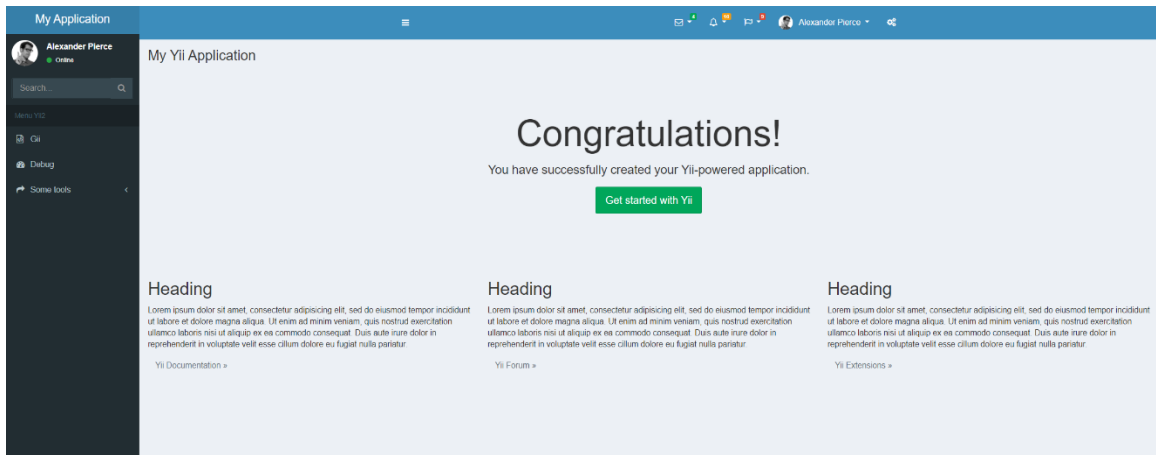
等待命令完成就下载好了模板。

然后是整合模板，我们把 `\advanced\vendor\dmstr\yii2-adminlte-asset\example-views\yiisoft\yii2-app` 目录下的 `layouts` 与 `site` 文件夹直接与 `\advanced\backend\views` 目录下的同名文件夹合并，重名文件替换就可以成功更改样式，结果如图：



我们发现 `layout` 的 `header` 部分显示出来效果不是很理想，于是一番研究后修改了 `\advanced\backend\views\layouts\header.php` 文件中第 20 行，把 `<ul class="nav navbar-nav">` 改为了 `<ul class="nav navbar-nav" style="display:inline;">`

修改后效果如下：



然后我们在模板的基础上进行修改，将网页调整为适配我们目的功能的样子，如下：



更改完样式后并没有相应功能的实现，尤其是权限部分，所以我们下一步需要添加代码实现权限管理。由于我们的网址是不断有新用户进行注册的，所以我们不使用存取控制过滤器（ACF）这种静态的权限管理系统而使用动态的 RBAC 权限管理系统。

我们可以使用 **PhpManager**——通过 PHP 脚本存放授权数据，也可以使用 **DbManager**——通过数据库存放授权数据。考虑到用户数量，我们使用 **DbManager** 进行授权数据的存储。我们把 `\advanced\common\config\main.php` 文件进行修改，修改如下：

```
'components' => [  
    'cache' => [  
        'class' => 'yii\caching\FileCache',  
    ],  
    "authManager" => [  
        "class" => 'yii\rbac\DbManager',  
    ],  
],  
];
```

这样我们就可以使用 DbManager 了。

由于我们的权限层次是静态的不会改变，所以我们可以一开始就在代码里进行权限层次的初始化。具体操作如下：

我们在\advanced\console\controllers 目录下新建一个文件：RbacController.php，在里面加入以下代码：



console > controllers > RbacController.php

```
1  <?php
2  namespace console\controllers;
3
4  use Yii;
5  use yii\console\Controller;
6
7  class RbacController extends Controller
8  {
9      public function actionInit()
10     {
11         $auth = Yii::$app->authManager;
12         $auth->removeAll();
13
14         // 添加 "makeComment" 权限
15         $makeComment = $auth->createPermission('makeComment');
16         $makeComment->description = 'can make a comment';
17         $auth->add($makeComment);
18
19         // 添加 "deleteComment" 权限
20         $deleteComment = $auth->createPermission('deleteComment');
21         $deleteComment->description = 'can delete a comment';
22         $auth->add($deleteComment);
23
24         // 添加 "controlAuthority" 权限
25         $controlAuthority = $auth->createPermission('controlAuthority');
26         $controlAuthority->description = 'can control authority';
27         $auth->add($controlAuthority);
28
29         // 添加 "author" 角色并赋予 "makeComment" 权限
30         $author = $auth->createRole('author');
31         $auth->add($author);
32         $auth->addChild($author, $makeComment);
33
34         // 添加 "admin" 角色并赋予 "deleteComment"
35         // 和 "author" 权限
36         $admin = $auth->createRole('admin');
37         $auth->add($admin);
38         $auth->addChild($admin, $deleteComment);
39         $auth->addChild($admin, $author);
40
41
42         // 添加 "boss" 角色并赋予 "controlAuthority"
43         // 和 "author" 、"admin" 权限
44         $boss = $auth->createRole('superboss');
45         $auth->add($boss);
46         $auth->addChild($boss, $controlAuthority);
47         $auth->addChild($boss, $author);
48         $auth->addChild($boss, $admin);
49
50
51         // 为用户指派角色。其中 1 和 2 是由 IdentityInterface::getId() 返回的id
52         // 通常在你的 User 模型中实现这个函数。
53         //$auth->assign($author, 2);
54         //$auth->assign($admin, 1);
55     }
56 }
```

完成后我们可以通过 cmd 使用 `yii rbac/init` 命令初始化权限相关的四张表：

**itemTable:** 该表存放授权条目即角色和权限，默认表名为 "auth\_item"。

**itemChildTable:** 该表存放授权条目的层次关系，默认表名为 "auth\_item\_child"。

**assignmentTable:** 该表存放授权条目对用户的指派情况，默认表名为 "auth\_assignment"。

**ruleTable:** 该表存放规则，默认表名为 "auth\_rule"。

这里简单介绍以下这四张表：

权限是最小单元。

角色可以认为是权限的集合，一个角色可以拥有一个权限，也可以拥有几个权限。  
角色可以从其他角色处继承他所拥有的权限。

用户可以与权限建立映射关系。

规则可以给角色和权限增加额外的约束条件。

我们希望注册用户默认拥有留下评论的权限，即对应 `author` 角色，所以我们在前台代码用户注册相关的部分进行修改：

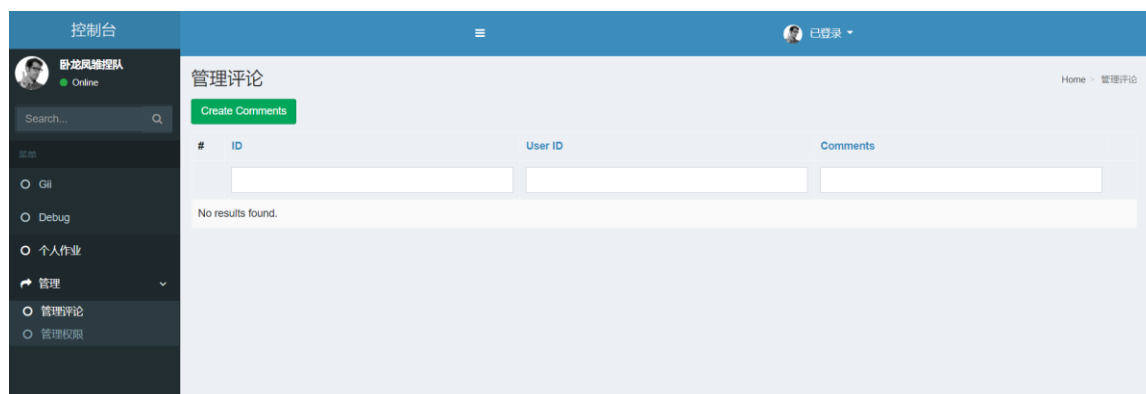
```
public function signup()
{
    if (!$this->validate()) {
        return null;
    }

    $user = new User();
    $user->username = $this->username;
    $user->email = $this->email;
    $user->setPassword($this->password);
    $user->generateAuthKey();
    $user->generateEmailVerificationToken();

    $user->save();
    //设置默认权限为"author"
    $auth = \Yii::$app->authManager;
    $authorRole = $auth->getRole('author');
    $auth->assign($authorRole, $user->getId());

    return $this->sendEmail($user);
}
```

然后我们希望有管理员不仅能够进行评论，还可以对评论进行管理，于是同前台的操作，我们通过 **gii** 自动生成评论表的管理页面：

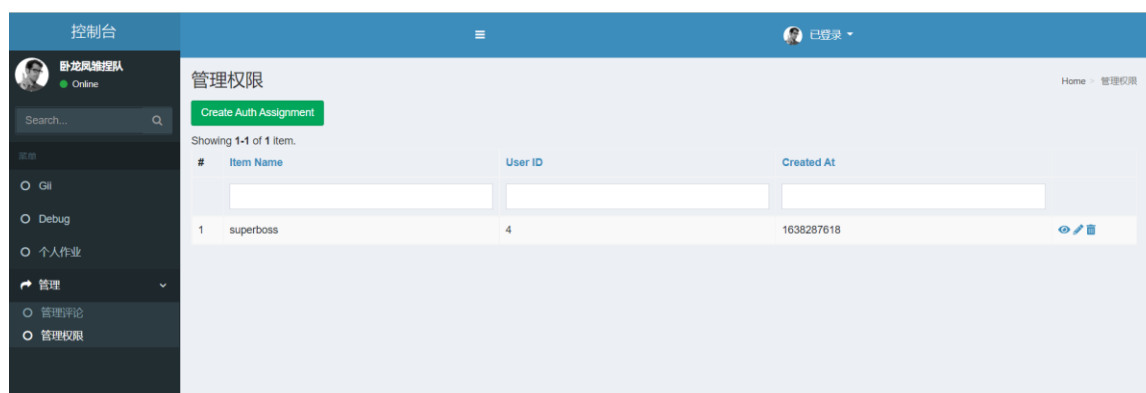


然后修改左部的菜单栏，让“管理评论”按钮可以指向这个评论表的管理页面，通过让该按钮只对管理员显示：

```
'items' => [
    ['label' => '管理评论', 'url' => ['comments/index'], 'visible' => Yii::$app->user->can('deleteComment')],
```

随后我们希望有一个超级管理员，他能够对权限进行管理，比如把一个普通用户升级为管理员或者把一个管理员降级为普通用户。

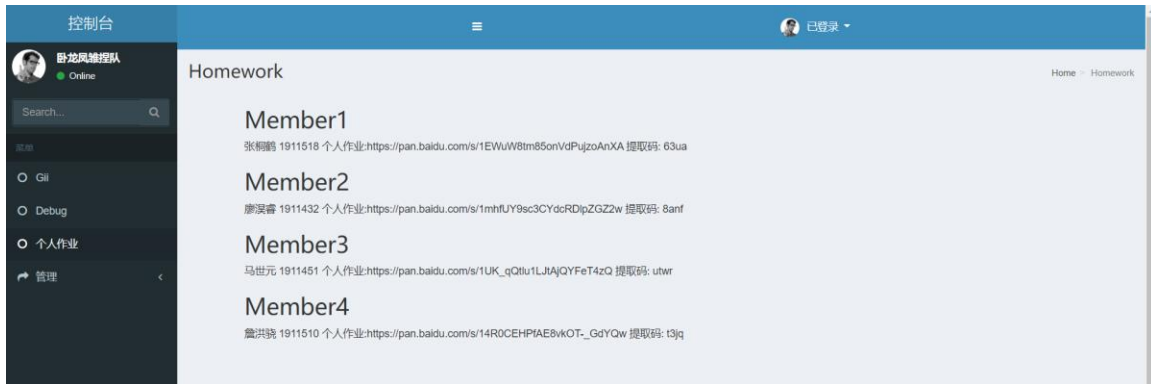
于是我们照猫画虎，再次用 **gii** 生成权限管理表（上文提到的 **auth\_assignment**）的页面。



同样的修改左部菜单栏，让“管理权限”按钮可以指向这个权限管理表的管理页面，通过让该按钮只对超级管理员显示：

```
['label' => '管理权限', 'url' => ['authassignment/index'], 'visible' => Yii::$app->user->can('controlAuthority')],
```

最后制作一个网页，用于存放我们小组的个人作业百度网盘链接。



然后绑定左边的菜单栏的“个人作业”按钮，再在存取控制过滤器（ACF）中增加访问该页面的权限（路径\advanced\backend\controllers\SiteController.php）。

```
['label' => '个人作业', 'url' => ['site/homework']],
```

```
class SiteController extends Controller
{
    /**
     * {@inheritdoc}
     */
    public function behaviors()
    {
        return [
            'access' => [
                'class' => AccessControl::className(),
                'rules' => [
                    [
                        'actions' => ['login', 'error'],
                        'allow' => true,
                    ],
                    [
                        'actions' => ['logout', 'index'],
                        'allow' => true,
                        'roles' => ['@'],
                    ],
                    //添加访问个人作业的权限
                    [
                        'actions' => ['logout', 'homework'],
                        'allow' => true,
                        'roles' => ['@'],
                    ],
                ],
            ],
        ],
    }
}
```

至此后台的全部工作完成。