

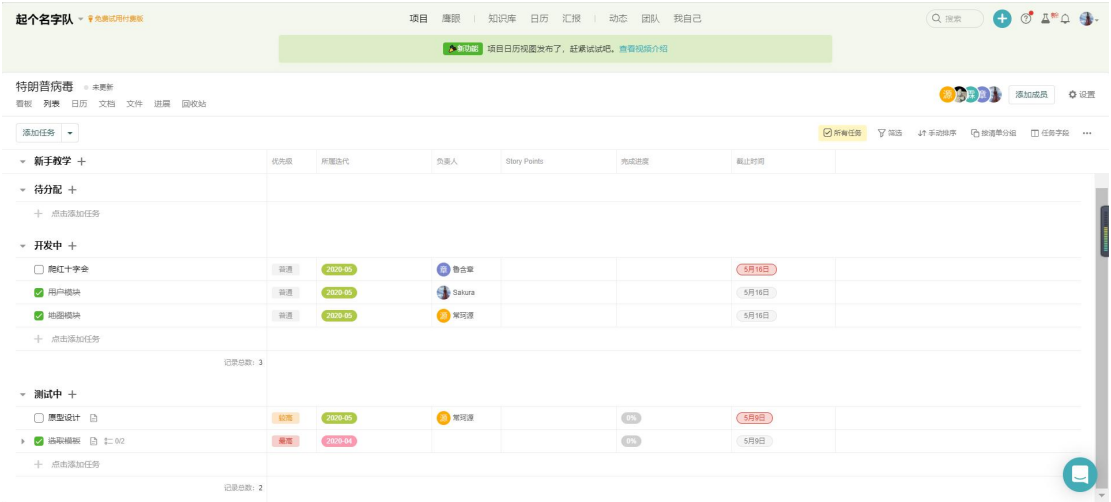
# 实现文档

## 一、任务分工过程

这里提供一些 Tower 截图，所有截图的进度截止到 2020/6/13 12:00，到此时还有部分模块尚未完成。

各模块的分工如下：

- 爬红十字会(frontend)：鲁含章；
- 用户模块(frontend)：董佳霖、孙一丁；
- 地图模块(backend)：常珂源、翟江天。



文件					
上传文件					
名称	大小	最后修改时间	最后版本上传者		
 20200612数据字典-2.xlsx	14K	昨天	Sakura		<a href="#">下载</a> <a href="#">重命名</a> <a href="#">更多</a>
 需求文档 20200612-1.docx	32.5K	昨天	Sakura		
 20200612数据字典 - 副本.xlsx	14K	昨天	Sakura		
 20200608-3数据字典.xlsx	13.7K	6月9日	Sakura		
 设计文件 草稿.docx	174K	5月5日	翟江天		
 设计文件 草稿.pdf	126K	5月5日	翟江天		
 需求文档 草稿.docx	23.9K	5月4日	Sakura		
 需求文档 草稿.pdf	224K	5月4日	Sakura		
 IEEE830.pdf	373K	4月20日	鲁含章		
 会议记录.docx	14.4K	4月18日	鲁含章		

附件中心

## 项目进展



## 二、Git 提交记录

Github 仓库: [https://github.com/yoko2001/tb\\_yii2020/](https://github.com/yoko2001/tb_yii2020/)

为了避免 Git Merge 带来的奇奇怪怪的错误, 我们采用的是单人本地 Merge, 再 Commit 的方式。

注意: 因为考虑到从 Github 仓库 clone 时速度过慢, 仓库中的文件不包括相对较大且重复的 vendor 文件, 可以通过 yii2 框架复制到根目录下。

Github 提交记录截图 (截止到 2020/6/13 12: 00):

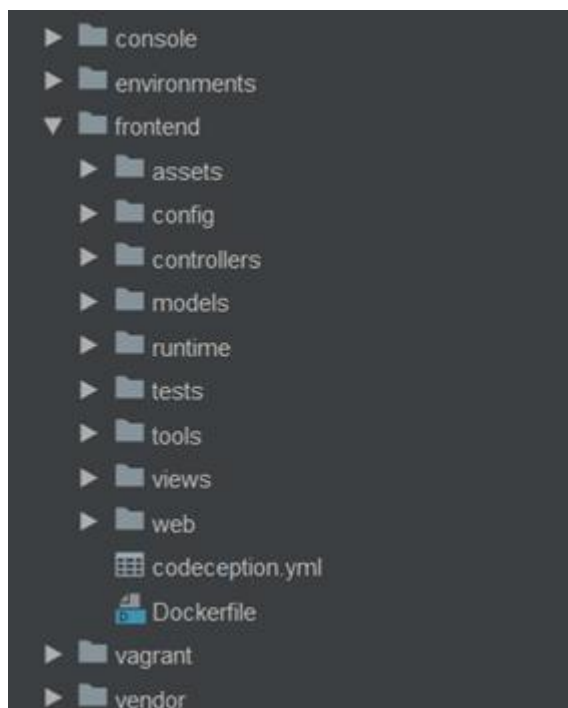


Commits on Jun 2, 2020		
add demand form yoko2001 committed 11 days ago	bc9023c	<>
now can view by smaller pages, and use delete button yoko2001 committed 11 days ago	bd2eb51	<>
realize add using atlantis's form and table,(backend) yoko2001 committed 11 days ago	f11f5b0	<>
Commits on May 17, 2020		
add basic migrates, fix logout bug in backend yoko2001 committed 27 days ago	2381392	<>
Commits on May 16, 2020		
add standard deployment doc yoko2001 committed 28 days ago	5ba992b	<>
fix logout yoko2001 committed 28 days ago	3e0afd7	<>
test migrate yoko2001 committed 28 days ago	477b497	<>
Commits on May 15, 2020		
first commit yoko2001 committed 29 days ago	31e6b53	<>

### 三、实现思路及主要代码

#### I. 前台 frontend

前台整体代码目录展示



##### 1. 创建前台模板

我们选用 absolute 模板，首先分别将

- 1)static 文件夹放到 frontend\web 目录下
- 2)main\_layout 放到 frontend\views 目录下
- 3)AppAsset 放到 frontend\assets 目录下

在之后 frontend 目录下所有的 controllers 类中都加上

```
public $layout = "main_layout";
```

这样所有前台的页面都应用了我们选用的 absolute 模板

## 2.信息管理模块

### 1) 疫情数据展示

我们从 <https://github.com/BlankerL/DXY-COVID-19-Crawler> 下载项目中 DXYArea.csv 文件

在实现.csv 到.sql 文件的转换时，我们采用利用 python 作为中间语言实现。

表头部分的代码手写完成。

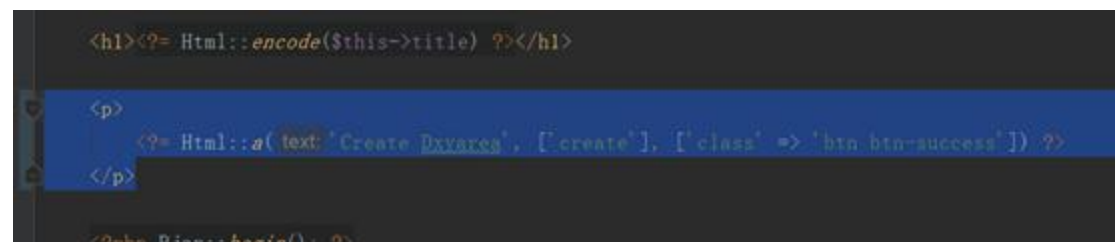
插入部分的代码：

- 1) 在 python 中读取 csv 文件的每一行
- 2) 在控制台打印目标的 sql 语句

得到 sql 文件后，我们在 phpmyadmin 里导入 DXYArea 表。

之后利用 gii 中的 models 和 CRUD 生成关于本小节展示世界疫情数据的 mvc 文件。

在 dxyarea/view/index.php 里删除用于展示按钮及修改栏的控件代码。



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

<p>
    <?= Html::a( text: 'Create Dxyarea', ['create'], ['class' => 'btn btn-success']) ?>
</p>

<?php $jax->begin(); ?>
```

```
<?php // echo $this->render('_search', ['model' => $searchModel]); //  
  
<?= GridView::widget([  
    'dataProvider' => $dataProvider,  
    'filterModel' => $searchModel,  
    'columns' => [  
        ['class' => 'yii\grid\SerialColumn'],  
  
        'id',  
        'locationId',  
        'continentName',  
        'continentEnglishName',  
        'countryName',  
        //'countryEnglishName',  
        //'countryFullName',  
    ],  
)
```

之后，页面剩下的部分就构成了单纯用于展示的表格信息  
修改用于显示的列：

```
        'id',  
        //' locationId',  
        //' continentName',  
        //' continentEnglishName',  
        'countryName',  
        //' countryEnglishName',  
        //' countryFullName',  
        //' provinceName',  
        //' provinceEnglishName',  
        //' provinceShortName',  
        'currentConfirmedCount',  
        //' confirmedCount',  
        //' suspectedCount',  
        'curedCount',  
        'deadCount',  
        //' comment',  
        //' cities',  
        //' updateTime',
```

## 2) 用户反馈模块

在 phpmyadmin 中创建 contactme 表。

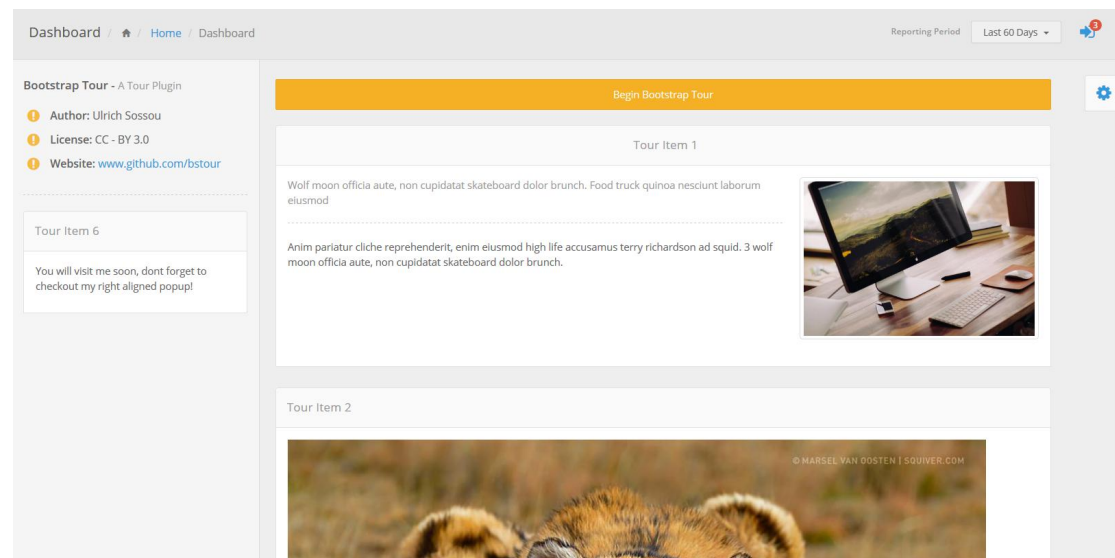
之后利用 gii 中的 models 和 CRUD 生成关于本小节展示世界疫情数据的 mvc 文件。

最后，在 main\_layout 里将左边的“用户反馈”按钮的链接设置一下。

```
<li>
  <a class="accordion-toggle" href="index.php?r=contactme/create">
    <span class="glyphicon glyphicon-fire"></span> 用户反馈</a>
</li>
```

## 3.主页模块

这里的主页套用了 absolute 里的 Site Tour。



使用 F12 选择器，将我们需要内容从源代码中摘取下来，放到主页的 index.php 中,其所需的依赖从源代码的头文件中查看，然后将其放到 web 文件夹中，并在 APPAssets.php 中添加路径即可。

```
<div class="navbar-branding">
  <a class="navbar-brand" href="index.php">
    <b>疫情</b>物资
  </a>
  <span id="toggle_sidemenu_1" class="ad ad-lines"></span>
</div>
```

由于使用的 main-layout 作为模板，所以改变其中的跳转操作，即可跳转到我们想要的模块。

这里直接跳转到了 web 文件夹下的 index.php 文件

#### 4. 累计确诊疫情分布图模块

这个模块的 php 文件在 views/site 文件夹内, 在 siteController 中需要添加渲染这个页面的函数, 触发事件进行操作。

主要的思想就是使用 echarts 这个开源的图表样式 js 文件。

由于使用的是网络上的实时数据, echarts 也可以直接引用为网站上在线的, 所以不需要在 Appasset 中导入我们需要的 js 文件等。

在 php 下的 HTML 文件的 header 中引入样式, 设定 dom 容器的格式, 以及地图的大小。

```
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>疫情地图展示</title>
<style>
    .container {
        width: 1200px;
        margin: 0 auto;
    }

    #myEcharts {
        width: 800px;
        height: 500px;
        border: solid 1px red;
        margin: 0 auto;
    }
</style>
<script src="https://www.echartsjs.com/examples/vendors/jquery/jquery.js"></script>
<!-- 引入 echarts.js -->
<script src="https://www.echartsjs.com/examples/vendors/echarts/echarts.min.js?v=1578305236132"></script>
<!-- 引入中国的地图数据js文件, 引入后会自动注册地图名字和数据-->
<script src="https://www.echartsjs.com/examples/vendors/echarts/map/js/china.js?v=1578305236132"></script>
```

在 html 的 body 中对图表按照 echarts 教程中进行绘制, 通过 key-value 键值对, 设定图表的配置。



```

option = {
  title: {
    text: '中国累计确诊疫情图',
    left: 'center'
  },
  tooltip: {
    trigger: 'item'
  },
  legend: {
    orient: 'vertical',
    left: 'left',
    data: ['中国疫情图']
  },
  visualMap: {
    type: 'piecewise',
    pieces: [
      { min: 1000, max: 1000000, label: '大于等于1000人', color: '#372a28' },
      { min: 500, max: 999, label: '确诊500-999人', color: '#4e160f' },
      { min: 100, max: 499, label: '确诊100-499人', color: '#974236' },
      { min: 10, max: 99, label: '确诊10-99人', color: '#ee7263' },
      { min: 1, max: 9, label: '确诊1-9人', color: '#f5bba7' },
    ],
    color: ['#E0022B', '#E09107', '#A3E00B']
  },
},

```

比如这里就设置了图表的标题，触发器类型，以及不同确诊人数的颜色等。

之后从 inews.qq 在线获取各省份实时数据，最后再使用我们指定的配置项和数据显示图表：



```

var province = res.areaTree[0].children;
for (var i = 0; i < province.length; i++) {
    var json = {
        name: province[i].name,
        value: province[i].total.confirm
    }
    newArr.push(json)
}
console.log(newArr)
console.log(JSON.stringify(newArr))
//使用指定的配置项和数据显示图表
myChart.setOption({
    series: [
        {
            name: '确诊数',
            type: 'map',
            mapType: 'china',
            roam: false,
            label: {
                show: true,
                color: 'rgb(249, 249, 249)'
            },
            data: newArr
        }
    ]
})

```

## 5. 团队/个人作业模块

这里就是建立了多个 php 文件，在 siteComtroller 中设置不同的触发函数，最后在 main-layout 文件中写入这些导致触发的时间即可。

下面就是 sitecomtroller 中的函数。

注意：第二个函数会传入 singlework 参数，之后在其 php 中会依次输出他们。

```

/**
 * display teamwork
 */
public function actionMainteamwork(){
    return $this->render('mainteamwork');
}
/**
 * display selfwork1
 */
public function actionSelfwork(){
    $singlework = [
        ['name' => 'Web前端初探', 'namecontent' => ''],
        ['name' => 'Web前端设计', 'namecontent' => ''],
        ['name' => '开源建站工具初试文档', 'namecontent' => ''],
    ];
    return $this->render('Selfwork', ['singlework' => $singlework]);
}

```

Main-layout 中触发事件的语句:

```

<li>
    <a href="<?php echo Url::to(['site/mainteamwork']) ?>">
        <span class="glyphicon glyphicon-flag"></span> 团队作业</a>
</li>

```

个人作业的 php 文件:

```

<h1>lhz</h1>

<p>
    所属成员: lhz ,NKU
</p>
<br/>
<table class="table table-bordered table-hover">
<thead>
    <tr>
        <th>作业名</th>
        <th>内容</th>
    </tr>
</thead>
<tbody>
<?php foreach ($singlework as $v) :?>
    <tr>
        <td><?=$v['name']?></td>
        <td><?=$v['namecontent']?></td>
    </tr>
<?php endforeach;?>
</tbody>
</table>

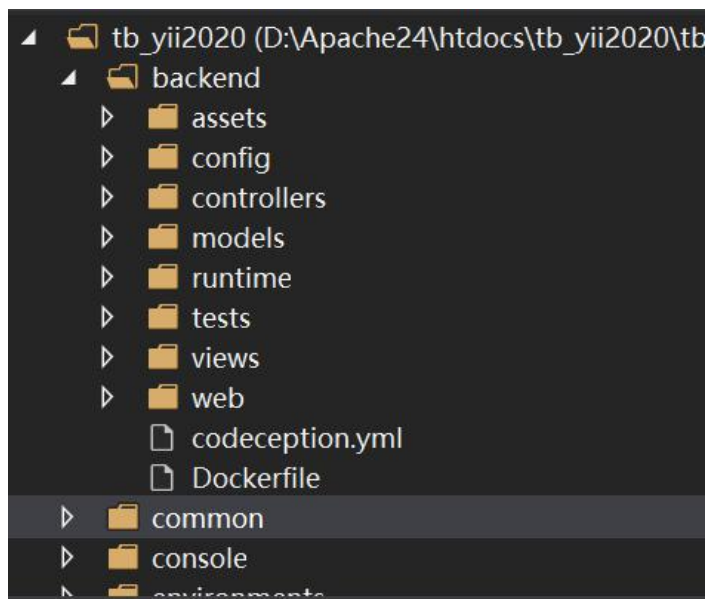
```

## 6. 前端模板使用时一些问题:

1. 在引入样式文件时一定要引入正确, 否则效果出不来, 而且还要在 Appasset 中把样式导入进去。具体的调试方法可以看原模板的 network, 看看引入的文件路径是什么, 然后再将其引入到 yii 中。
2. 改变页面的一些图标可以使用开发者工具找到源代码相应位置, 在 web 中导入想要的图标, 最后再改变 main-layout 即可。

## II. 后台 backend

### a. 后台整体代码目录展示

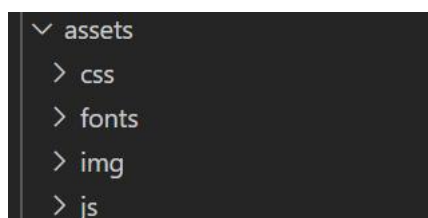


## b. 后台模板更换

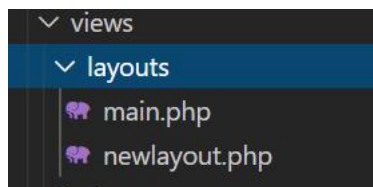
我们选用的 Atlantic 模板。这个模板有很多方便的交互式的组件，可以实现有趣的功能。

更换步骤：

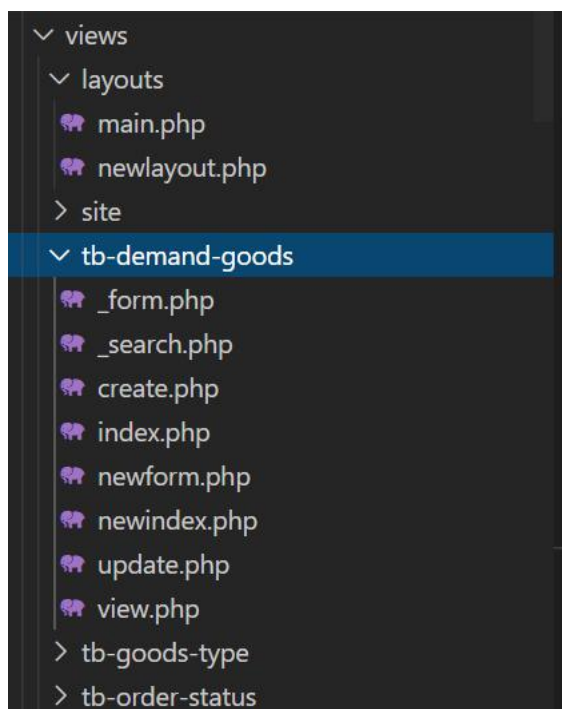
将 Atlantic 模板的 assets 全部内容复制到 backend 的 assets 中



将 Atlantic 的 index.php 中 <content>部分拿掉，另存为新的 newlayout.php。



遇到需要使用的模块就将这个模块进行修改，放入 view 中对应的类别管理。



以 tb-demand-goods 为例，除了 gii 自动生成的页面，其他以 new 开头的页面都采用 atlantis 的样修改而成的。几乎都支持从数据库读取并动态生成页面内容。

最后由于出现了 jquery 冲突等问题，禁用掉了原 yii 的部分才解决。

## c. 各模块实现

### 1. tb-demand-goods

#### 1.1 主要实现思路

模块名称：tb-demand-goods

主要编写者：董佳霖

功能：

1. 进行需求信息的展示，以美观的表格的形式
2. 进行需求信息的插入与删除，反馈在信息展示页面上
3. 进行需求信息的填报。通过填写一个表单来完成。表单必须易于使用且免维护。需要自动填入一些确定的信息（例如 id）

实现过程：

0. 这个模型是比较早搭建的，虽然实现起来不是特别复杂，但是摸索了一些经验。

1. 首先通过第一版的数据库设计，进行数据库的搭建。在中期我们学习了如何编写 migrate 文件，整个过程变得方便很多。

2. 利用 yii 自带的 gii 功能，根据课上讲授的 model generate 方法，直接生成了带有数据库操作的 model

3. 修改我们所使用的 Atlantic 模板，制成了数据展示 newindex.php 以及数据表单填写 newform.php 两个页面。使用 common/tool/tool class 中的 echo 函数动态将数据写入表单。并完成了数据交互的功能。该 model 中进行数据交互主要利用的是 yii 生成的 model 和 modelquery 类。整体风格比较易读，两个页面代码由于动态生成，也比较简洁。在表单填写中，采取的是使用 post 指令传输多条数据，然后将数据反馈给数据操作 model 进行插入的模式。并且为了使用方便，在开发后期加入了自动填写用户名等方式。

实现难点：

主要的困难便是数据交互部分以及页面动态生成部分。首先花费了不少时间理解 mvc 的结构，再去学习如何动态生成页面，再学习 controller 的正规用法，等等都构成了很大困难。一开始插入与删除这两个本来应该由 tbdemandgoodscontroller 完成的内容，由于结构的不熟悉，放在了 sitecontroller 中，直到比较靠后的版本才将其改正。

另外一个难点就是 css 冲突和 jquery 冲突，也是经过了很久的调试才最终能够使用全部功能。例如数据展示区域的翻页功能，就经过了很多次的试验，才最终完成。

## 1.2 主要代码截图

### 1.2.1 动态生成方法

所有的动态生成方法被放置在工具类/common/tool/tool 中，例如生成一个带编辑与删除按钮的表单项，采用的 tool 为：

```

public static function echoAllRecordsWithEdit($array, $request){
    $totalnum = count($array);
    for ($i=0; $i < $totalnum; $i++) {
        $thearray = $array[$i];
        $suid = ArrayHelper::getValue($thearray,'tb_dgUser', $default = true);
        $dId = ArrayHelper::getValue($thearray,'tb_dgId', $default = true);
        $dTypeid = ArrayHelper::getValue($thearray,'tb_dgType', $default = true);
        $dTypeArr = TbGoodsType::findOne($dTypeid);
        $dType = ArrayHelper::getValue($dTypeArr,'tb_gtName', $default = true);
        $dPrice = ArrayHelper::getValue($thearray,'tb_dgPrice', $default = true);
        $dNum = ArrayHelper::getValue($thearray,'tb_dgNum', $default = true);
        $dRemark= ArrayHelper::getValue($thearray,'tb_dgRemark', $default = true);
        echo tool::getAllRecordsWithEdit($request, $suid, $dId, $dType, $dNum, $dPrice
    }
}

public static function echoAllDmTypes(){
    $dTypeArr = TbGoodsType::find()->all();
}

```

```

public static function getAllRecordsWithEdit($request, $suid, $dId, $dType, $dNum, $dPrice, $dRemark){
    $ret =<<<EOF
    <tr>
        <td>$suid</td>
        <td>$dId</td>
        <td>$dType</td>
        <td>$dPrice</td>
        <td>$dNum</td>
        <td>$dRemark</td>
        <td>
            <div class="form-button-action">
                <button type="button" data-toggle="tooltip" title="" class="btn btn-link btn-primary btn-
                <i class="fa fa-edit"></i>
                </button>
                <form action="/tb_yii2020/tb_yii2020/backend/web/index.php?r=tb-demand-goods%2Fdelete" me
                <input type="hidden" name="_csrf-backend" value="<?=$request->csrfToken ?>">
                <button name="pid" value=$dId type="submit" data-toggle="tooltip" title="" class="btr
                <i class="fa fa-times"></i>
                </button>
                </form>
            </div>
        </td>
    </tr>
    EOF;
    return $ret;
}

```

通过这样的方式就能直接将复杂的内容动态生成在页面上。

```

?>
<?php common\tool\tool::echoAllRecords($array1);?>
</tbody>

```

可以看到 delete button 会触发一个 action 到 controller，下面有 controller 的 actionDelete 函数



```

public function actionDelete()
{
    $id = \Yii::$app->request->post('pid', null);
    $delrow = TbDemandGoods::findOne($id);
    if($delrow->validate()){
        $delrow->delete();
        echo "<script>alert('del success')</script>";
    }else{
        echo "<script>alert('del fail')</script>";
    }
    return $this->redirect(['site/demand']);
}

```

直接查找这一列，进行完整性验证后删除，抛出消息证明是否删除成功。

### 1.2.2 提交表单的数据传输

表单不光需要发送信息，也需要提取必要信息：

例如：

```

<div class="form-group">
    <label for="basic-url">提交人</label>
    <div class="input-group mb-3">
        <div class="input-icon">
            <span class="input-icon-addon">
                <i class="fa fa-user"></i>
            </span>
            <input name = "UserID" type="text" class="form-control"
                disabled="disabled" placeholder="<? = Yii::$app->user->identity->username ?>">
        </div>
    </div>
</div>

```

直接选取了当前用户作为用户 id，不让用户自己填写。

```

form action = "/tb_yii2020/tb_yii2020/backend/web/index.php?r=tb-demand-goods%2Fadd"
method="post" name="addform"

```

整个表单是一个 form 套起来的。在最下方的按钮处，将 action 的发送与按钮连接。

```

<div class="card-action">
    <input type="button" id="addRowButton" value="Submit" class="btn btn-success"
        onclick="javascript:fsubmit(document.addform);" />
    <button class="btn btn-danger">Cancel</button>
</div>

```

其中 fsubmit 是自定义的 jquery 函数，主要就是把参数 submit 出去。仿照的其他网站处理方式。

## 2. tb-weal-goods

## 2.1 实现思路

模块名称：tb-weal-goods

主要编写者：董佳霖、孙一丁

功能：

1. 进行冗余物资信息的展示，以美观的表格的形式
2. 进行冗余物资信息的插入与删除，反馈在信息展示页面上
3. 进行冗余物资信息的填报。通过填写一个表单来完成。表单必须易于使用且免维护。需要自动填入一些确定的信息（例如 id）

实现过程：

1. 这个模型是基于 tbdemandgoods 模型搭建的
2. 网页设计上重新使用了 tbdemandgoods 中的表单和数据表。节省了时间，功能依然完备。
3. 重新对数据操作的 model 进行连接，使之能正常工作。

实现难点：

在同类型的 tbdemandgoods 基础上做一些修改，因为本质上内容差不多。所以相对容易一些。但是修改数据库需要重新生成，导致我们在利用 model 查找上花费了一些精力。

## 2.2 主要代码截图

(1) 路径为 common\tool\tool4.php

只展示了 tool4.php 的一部分，这部分代码被 newindex.php 调用，用于展示所有的赋予物资记录，并且添加编辑按钮。

```
public static function echoAllRecordsWithEdit($array, $request){
    $totalnum = count($array);
    for ($i=0; $i < $totalnum; $i++) {
        $thearray = $array[$i];
        $uid = ArrayHelper::getValue($thearray, 'tb_wgUser', $default = true);
        $did = ArrayHelper::getValue($thearray, 'tb_wgId', $default = true);
        $dtypeid = ArrayHelper::getValue($thearray, 'tb_wgType', $default = true);
        $dtypearr = TbGoodsType::findOne($dtypeid);
        $dtype = ArrayHelper::getValue($dtypearr, 'tb_gtName', $default = true);
        $dprice = ArrayHelper::getValue($thearray, 'tb_wgPrice', $default = true);
        $dnum = ArrayHelper::getValue($thearray, 'tb_wgNum', $default = true);
        $dremark = ArrayHelper::getValue($thearray, 'tb_wgRemark', $default = true);
        echo tool::getAllRecordsWithEdit($request, $uid, $did, $dtype, $dnum, $dprice, $dremark);
    }
}
```

调用上面的 tool4.php 部分的代码如下。

```

<div class="table-responsive">
    <table id="add-row" class="display table table-striped table-hover" >
        <thead>
            <tr>
                <th>UserID</th>
                <th style="width: 10%">WealID</th>
                <th style="width: 10%">WealType</th>
                <th style="width: 10%">Num</th>
                <th style="width: 10%">Price</th>
                <th>Detail</th>
                <th style="width: 10%">Action</th>
            </tr>
        </thead>
        <tfoot>
            <tr>
                <th>UserID</th>
                <th>WealID</th>
                <th>Weal
                    Type</th>
                <th>Num</th>
                <th>Price</th>
                <th>Detail</th>
                <th>Action</th>
            </tr>
        </tfoot>
        <tbody>
            <?php
                $array1 = $searchModel::find()->all();
            ?>
            <?php common\tool\tool4::echoAllRecordsWithEdit($array1, Yii::$app->request);?>
        </tbody>
    </table>
</div>

```

(2) 路径为 backend\controllers\TbWealGoodsController.php 下用于验证填入数据表单的正确性。其中，特别增添了用户输入自己 id 和昵称的两种验证方式。但在最终的版本中，为了避免恶意注入，默认修改为自动填入当前登录用户的 id。

```

public function actionAdd()
{
    echo "<script>alert(124)</script>";
    $addrow = new TbWealGoods();
    $request = Yii::$app->request;
    $uid = $request->post('UserID', null);
    if ($uid == "" or $uid == null){
        $addrow->tb_wgUser = Yii::$app->user->identity->id;
    }else{
        $addrow->tb_wgUser = $request->post('UserID', null);
    }
    $addrow->tb_wgType = $request->post('WType', null);
    if (!is_int($addrow->tb_wgType)){
        $arr = TbGoodsType::find()->all();
        $cnt = count($arr);
        for($i = 0; $i < $cnt; $i++){
            $tmp = ArrayHelper::getValue($arr[$i], 'tb_gtName', $default = true);
            if ($tmp == $addrow->tb_wgType ){
                $addrow->tb_wgType = ArrayHelper::getValue($arr[$i], 'tb_gtId', $default = true);
                break;
            }
        }
    }
    $addrow->tb_wgNum = $request->post('WNum', null);
    $addrow->tb_wgPrice = $request->post('WPrice', null);
    $addrow->tb_wgRemark = $request->post('WDetail', null);
    $addrow->tb_wgAddress = $request->post('WAddress', null);
    if($addrow->validate()){
        echo "<script>alert('success')</script>";
        $addrow->save();
    }else{
        echo "<script>alert($addrow->tb_wgType)</script>";
    }
    return $this->redirect(['site/weal']);
}

```

### 3. tb-bulletinboard

#### 3.1 实现思路

模块名称：tb-bulletinboard

主要编写者：孙一丁

功能：

1. 进行公告板的展示，以美观的 timeline 形式展示
2. 进行公告的发布，反馈在公告板页面上

实现过程：

1. 模型基于 gii 生成，以 tbdemandgoods 为基础搭建的
2. 网页的表单设计上重新使用了 tbdemandgoods 中的 newform.php 表单。
3. 公告板的展示参考了 Atlantis 中的 timeline 形式，在此基础上添加了包括显示发布时间、发布人、发布内容在内的元素。

实现难点：

在 Atlantis 的 timeline 模块的基础上, 由于要显示最近的五条消息, 当数据库中公告不足五条的时候, 将导致查找异常。所以为了实现的方便, 设置了最原始的库中就保有五条记录。

### 3.2 主要代码截图

(1) 位于 backend\views\Tb-Bulletinboard\newindex.php

为了展示最近的五条公告, 这里直接采用了在 html 中内嵌 sql 的方式, 简单方便。调用的 SYSDATE() 表示数据库系统当前的时间, 即计算公告发布时间以及系统当前时间的差值。

```
<div class="timeline-heading">
  <h4 class="timeline-title">
    Latest announcement
  </h4>
  <p><small class="text-muted"><i class="flaticon-message"></i>

  <?php
  echo "posted ";
  $posts = Yii::$app->db->createCommand(
    'SELECT timediff(SYSDATE(),tb_bPosttime) AS DIFF,tb_bUserId FROM tb_bulletinboard ORDER BY DIFF ASC limit 0,1'
  )->queryAll();
  if($posts[0]["DIFF"]<"838:59:59") echo $posts[0]["DIFF"];
  else echo "long long";
  echo " ago by ";
  echo $posts[0]["tb_bUserId"];
  ?>

  </small></p>
</div>
```

(2) 位于 backend\controllers\TbBulletinboardController 下

用于检查从 newindex.php 转过来的参数是否符合要求(调用 validate() 函数), 然后检查之后重定向到 site 的 showAction 下。

```
public function actionSss(){
    $addrow = new TbBulletinboard();
    $request = \Yii::$app->request;
    $addrow->tb_bUserId=$request->post("UserID",null);
    $addrow->tb_bContent=$request->post("Content",null);
    $nowtime=Yii::$app->db->createCommand('SELECT SYSDATE() AS nowtime')->queryAll();
    $addrow->tb_bPosttime=$nowtime[0]['nowtime'];
    if($addrow->validate()){
        $addrow->save();
    }else{
        echo "<script>alert('failed add')</script>";
    }
    return $this->redirect(['site/show']);
}
```

## 4. tb-orders 相关

### 4.1 实现思路

模块名称: tb-orders/tb-order-type



主要编写者：孙一丁、董佳霖

功能：

1. 进行现有订单的展示，以数据表的形式展示；
2. 管理员进行订单的添加，以及订单状态的编辑；

实现过程：

1. 模型基于 gii 生成，在 tbdemandgoods 的架构基础上进行了修改。
2. 订单的状态编辑表调用 bootstrap 包，编辑表单借用了 tb-demand-goods 中的 add 表单。
3. 重新对数据操作的 model 进行连接，使之能正常工作。

实现难点：

订单的状态编辑表单由于订单号、富裕物资号、需求物资号都是固定的，不能被用户更改，所以如何保留原有上述三项数据的情况下进行传参就是一个很大的问题。最后调用的 bootstrap 解决了这个问题。

为了在网页中展示出的不是订单状态号，而是订单状态号对应的订单状态文字，这里还引入了 tb-order-type 在查询的时候做笛卡尔积（或自然连接）。

在订单添加表单中，为了根据用户选择的物资类型展示出所有的需求物资号和富裕物资号，在 order-process.php 中采用了根据用户选择的物资类型展示出对应的数据表。

## 4.2 主要代码截图

(1) 位于 backend\views\tb-orders\orderprocess.php

这一部分代码片段，主要是为了实现根据用户选择的物资类型展示出对应的数据表，一开始所有的数据表都是 display:none 的状态，根据用户在下拉框中选择出的物资类型再进行对应的展示。

```

<div id="hiddenel1" style="display:none">
    <div class="col-md-12">
        <div class="card">
            <div class="card-header">
                <h4 class="card-title">对应类型的需求库信息</h4>
            </div>
            <div class="card-body">
                <div class="table-responsive">
                    <table id="basic-datatables" class="display table table-striped table-hover" >
                        <thead>
                            <tr>
                                <th>UserID</th>
                                <th>DemandID</th>
                                <th>Num</th>
                                <th>Price</th>
                                <th>Detail</th>
                            </tr>
                        </thead>
                        <tfoot>
                            <tr>
                                <th>UserID</th>
                                <th>DemandID</th>
                                <th>Num</th>
                                <th>Price</th>
                                <th>Detail</th>
                            </tr>
                        </tfoot>
                        <tbody>
                            <?php
                                $sql="SELECT * FROM tb_demand_goods WHERE tb_dgType=1";
                                $array=Yii::$app->db->createCommand($sql)->queryAll();

                                foreach($array as $value){
                                    $var1=$value["tb_dgUser"];
                                    $var2=$value["tb_dgId"];
                                    $var4=$value["tb_dgPrice"];
                                    $var5=$value["tb_dgNum"];
                                    $var6=$value["tb_dgRemark"];

                                    $ret =<<<<EOF
                                    <tr>
                                        <td>$var1</td>
                                        <td>$var2</td>
                                        <td>$var4</td>
                                        <td>$var5</td>
                                        <td>$var6</td>
                                    </tr>
                                }
                            </?php>
                        </tbody>
                    </table>
                </div>
            </div>
        </div>
    </div>
</div>

```



```

</script>
<script language="javascript">
    function fetchelement(obj){
        if(obj.value==1){/"食品"
            document.getElementById('hiddenele1').style.display = "inline";
            document.getElementById('hiddenele2').style.display = "none";
            document.getElementById('hiddenele3').style.display = "none";
            document.getElementById('hiddenele4').style.display = "none";
            document.getElementById('hiddenele5').style.display = "none";
            document.getElementById('hiddenele6').style.display = "none";
            document.getElementById('hiddenele7').style.display = "none";
        }
        else if(obj.value==2){
            document.getElementById('hiddenele1').style.display = "none";
            document.getElementById('hiddenele2').style.display = "inline";
            document.getElementById('hiddenele3').style.display = "none";
            document.getElementById('hiddenele4').style.display = "none";
            document.getElementById('hiddenele5').style.display = "none";
            document.getElementById('hiddenele6').style.display = "none";
            document.getElementById('hiddenele7').style.display = "none";
        }
        else if(obj.value==3){
            document.getElementById('hiddenele1').style.display = "none";
            document.getElementById('hiddenele2').style.display = "none";
        }
    }

```

特别介绍一下，弹出的模态窗和主窗口信息传输的部分。  
窗口：

```

<div class="modal-body">
    <p class="small">管理员权限：订单状态修改</p>
    <form action = "/tb_yii2020/tb_yii2020/backend/web/index.php?r=tb-orders%2Fchange" method="post">
        <input type="hidden" name="_csrf-backend" value="<?= Yii::$app->request->csrfToken ?>">
        <div class="row">
            <div class="col-md-6 pr-0">
                <div class="form-group form-group-default">
                    <label>orderid</label>
                    <input id="oid" type="text" name = "id" class="form-control" placeholder="">
                </div>
            </div>
            <div class="col-md-6 pr-0">
                <div class="form-group">
                    <label for="exampleFormControlSelect1">修正状态</label>
                    <select name = "status" class="form-control" id="changetype">
                        <option></option>
                        <?php \common\tool\tool::echoAllOTypes()??>
                    </select>
                </div>
            </div>
        </div>
    </form>
</div>

```

可以看到也使用了动态生成的下拉框，整体被 post action 包裹  
由于这个orderid是从主窗口传来的，所以我们在模态框窗口 show  
消息触发后，进行一些数据的转移。

```
<script language="javascript">
    $('#addRowModal').on('show.bs.modal', function(e){
        $('#oid').val($(e.relatedTarget).data("oid"));
    })
</script>
```

接收数据

```
public static function getAllRecords($oID, $oSellID, $oBuyID, $ostatus){
    $ret = <<<EOF
    <tr>
        <td>$oID</td>
        <td>$oSellID</td>
        <td>$oBuyID</td>
        <td>$ostatus</td>
        <td>
            <div class="form-button-action">
                <button type="button" data-oid= $oID data-toggle="modal"
                    data-target="#addRowModal" title="" class="btn btn-link btn-primary btn-lg"
                    data-original-title="Edit Task">
                    <i class="fa fa-edit"></i>
                </button>
            </div>
        </td>
    </tr>
    EOF;
    return $ret;
}
```

生成动态表单项的时候，button 被设置会传输 oID 信息。  
最终效果可以看到非常 amazing。就是排版差了点。