一、项目目录

代码E:\www\_jsga\data\code\will\jsga.wiiqq.com下面：

(这个目录是基于yii框架的)

builder：存放项目脚本，主要关注其中sql、和es即可

commands：文件夹内TaskController为应用常驻脚本（定时任务）;

config：配置文件，production.conf.php（生产环境） development.conf.php(开发环境)，test.conf.php（测试环境），apache或者nginx配置指定环境变量

filters：放置的为过滤权限的文件文件名对应模块名，文件夹内的其他文件为各个环境下的配置文件；

Models：文件夹内放置的为项目实体模型，其中task文件夹为具体定时任务实体类；

Runtime：文件夹放置项目运行时的日志信息；

Web：index.php根目录 下面的assets会存放wau加载时的临时文件

**重要：runtime文件夹 和web/assets文件夹要给777权限**

二、设置中心

设置中心即权限控制，基于：用户－角色－权限

(权限可以关联到市局和分局)

E:\www\_jsga\data\code\will\jsga.wiiqq.com\controllers\ BaseController：后台所有需要进行权限控制的controller都继承此类

控制器： \will\jsga.wiiqq.com\modules\admin\controllers

model类： \will\jsga.wiiqq.com\models\admin

ajax请求类：\will\jsga.wiiqq.com\modules\api\controllers\admin

views： \will\jsga.wiiqq.com\modules\admin\views

权限过滤： \will\jsga.wiiqq.com\config\filters；权限过滤走filter，每个模块都有各自的过滤器的配置文件，例如用户模块的 admin.yml，角色模块的role.yml

登录(url: /admin/admin/login)：

\will\jsga.wiiqq.com\modules\api\controllers\admin

\will\jsga.wiiqq.com\models\admin

/\*

YII框架代码解释：

**A、应用结构**



1、config 是配置文件

          ---  console.php

          ---  db.php    数据库配置

          ---  params.php  参数配置？

          ---  web.php    组件、ID

2、controllers 控制器

3、models  模型

4、runtime 缓存文件及log

5、vendor 核心文件

6、view   视图

          --- layouts   公共的文件

          --- site

7、web   入口文件及其js，css等

上面还要有一个modules文件夹，来放模块

**B、控制器**

Yii2把所有的控制器放在controllers 里面，控制器命名规则：

名称 + Controller.php，例如TestController.php，TestController文件就是TestController.php来自于Controller文件夹里面

每个类都必须有命名空间，命名空间就是文件存放的目录路径

<?php

namespace app\controllers;

//app 类似当前项目（网站）的根目录，网站的根目录（开辟的空间？）

//controllers 则是存放Controller 的目录

use yii\web\Controller;

//引用Controller

//引用类的规则命名空间+ 类名

//yii的所有控制器都是引用这个yii\web\Controller类

//在vendor\yiisoft\yii2\web里面（如何引用到这里？）

class TestController extends Controller{

public function actionIndex()

//Index 的话是默认路由

//访问的路由index.php?r=test 或者index.php?r=test/index

//这里要给外部访问的都必须加上action

{

echo ‘Hello Word’;

}

public function actionShowUser(){

//访问的路由index.php?r=test/show-user

//如果写成index.php?r=test/showUser 是错误

//名字带驼峰的

echo ‘ShowUser’;

}

}

?>

Yii 2.0 使用 [yii\web\Controller](http://www.yiichina.com/doc/api/2.0/yii-web-controller) 作为控制器的基类，这些变化最明显的影响是，当你在写控制器操作的代码时， 应该返回（return）要渲染的内容而不是输出（echo）它：

public function actionView($id)

{

$model = \app\models\Post::findOne($id);

if ($model) {

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

} else {

throw new \yii\web\NotFoundHttpException;

}

}

**C、视图**

视图的文件全部放在 views 目录中，layout放的是公共文件，对应于控制器中的test，在views里面新建一个test文件夹，然后控制器名称小写作为子目录 , 接下来就是方法名.php例如 TestController中的index方法，则创建后的文件目录应该为views/test/index.php

$this->render() 和 $this->renderPartial()都可以调用视图文件，两个的区别在于 render的时候会调用 lagout 的公共的文件(如 views/layouts/main.php)

<?php

namespace app\controllers;

use yii\web\Controller;

class TestController extends Controller{

public function actionIndex(){

//这里注意 return 是必须的, 不然不会返回视图

//index 是调用视图的文件

//数组里面的 data 则是注入到视图里面的数据

return $this->render(‘index’ , [‘data’=>[]]);

//或者

//return $this->renderPartial(‘index’ , [‘data’=>[]]);

}

}

?>

视图内的特殊变量 $this 不再指向当前控制器或小部件， 而是指向**视图**对象，**视图**对象为 [yii\web\View](http://www.yiichina.com/doc/api/2.0/yii-web-view) 的实例，他代表了 MVC 模式中的视图部分。 如果你想要在视图中访问一个控制器或小部件，可以使用 $this->context。

**D、命名空间**

当前命名方案与目录结构相吻合，例如，yii\web\Request 就表明对应的类文件是 Yii 框架文件夹下的 web/Request.php 文件。

组件类类拆分成了两个类：[yii\base\Object](http://www.yiichina.com/doc/api/2.0/yii-base-object) 和 [yii\base\Component](http://www.yiichina.com/doc/api/2.0/yii-base-component)。 [Object](http://www.yiichina.com/doc/api/2.0/yii-base-object) 类是一个轻量级的基类，你可以通过 getters 和 setters 来定义 [对象的属性](http://www.yiichina.com/doc/guide/2.0/concept-properties)。[Component](http://www.yiichina.com/doc/api/2.0/yii-base-component) 类继承自 [Object](http://www.yiichina.com/doc/api/2.0/yii-base-object)， 同时进一步支持 [事件](http://www.yiichina.com/doc/guide/2.0/concept-events) 和 [行为](http://www.yiichina.com/doc/guide/2.0/concept-behaviors)。

如果不需要用到事件或行为，应该考虑使用 [Object](http://www.yiichina.com/doc/api/2.0/yii-base-object) 类作为基类。 这种类通常用来表示基本的数据结构。

创建(声明)一个命名空间需要使用namespace关键字，当前脚本文件的第一个命名空间前面不能有任何代码

PHP支持两种抽象的访问当前命名空间内部元素的方法，\_\_NAMESPACE\_\_ 魔术常量和namespace关键字。

常量\_\_NAMESPACE\_\_的值是当前命名空间名称的字符串。在全局的，不包括在任何命名空间中的代码，它包含一个空的字符串。

关键字 namespace 可用来显式访问当前命名空间或子命名空间中的元素。它等价于类中的 self 操作符。

<?php

namespace A;

use B\D, C\E as F;

// 函数调用

foo(); // 首先尝试调用定义在命名空间"A"中的函数foo()

// 再尝试调用全局函数 "foo"

\foo(); // 调用全局空间函数 "foo"

my\foo(); // 调用定义在命名空间"A\my"中函数 "foo"

F(); // 首先尝试调用定义在命名空间"A"中的函数 "F"

// 再尝试调用全局函数 "F"

// 类引用

new B(); // 创建命名空间 "A" 中定义的类 "B" 的一个对象

// 如果未找到，则尝试自动装载类 "A\B"

new D(); // 使用导入规则，创建命名空间 "B" 中定义的类 "D" 的一个对象

// 如果未找到，则尝试自动装载类 "B\D"

new F(); // 使用导入规则，创建命名空间 "C" 中定义的类 "E" 的一个对象

// 如果未找到，则尝试自动装载类 "C\E"

new \B(); // 创建定义在全局空间中的类 "B" 的一个对象

// 如果未发现，则尝试自动装载类 "B"

new \D(); // 创建定义在全局空间中的类 "D" 的一个对象

// 如果未发现，则尝试自动装载类 "D"

new \F(); // 创建定义在全局空间中的类 "F" 的一个对象

// 如果未发现，则尝试自动装载类 "F"

// 调用另一个命名空间中的静态方法或命名空间函数

B\foo(); // 调用命名空间 "A\B" 中函数 "foo"

B::foo(); // 调用命名空间 "A" 中定义的类 "B" 的 "foo" 方法

// 如果未找到类 "A\B" ，则尝试自动装载类 "A\B"

D::foo(); // 使用导入规则，调用命名空间 "B" 中定义的类 "D" 的 "foo" 方法

// 如果类 "B\D" 未找到，则尝试自动装载类 "B\D"

\B\foo(); // 调用命名空间 "B" 中的函数 "foo"

\B::foo(); // 调用全局空间中的类 "B" 的 "foo" 方法

// 如果类 "B" 未找到，则尝试自动装载类 "B"

// 当前命名空间中的静态方法或函数

A\B::foo(); // 调用命名空间 "A\A" 中定义的类 "B" 的 "foo" 方法

// 如果类 "A\A\B" 未找到，则尝试自动装载类 "A\A\B"

\A\B::foo(); // 调用命名空间 "A\B" 中定义的类 "B" 的 "foo" 方法

// 如果类 "A\B" 未找到，则尝试自动装载类 "A\B"

在PHP中，别名是通过操作符 use 来实现的.

USE的两种用法，一个是导入命名空间，一个是use…as…，用于别名

**E、Yii的别名机制**

路径别名必须以 @ 符号开头， 以区别于普通文件目录路径或 URL。例如 @yii 就是指向 Yii 安装目录的别名。

别名用来表示文件路径和 URL，这样就避免了在代码中硬编码一些绝对路径 和 URL。

一个别名必须以 @ 字符开头，以区别于传统的文件路径和 URL。

别名 @yii 指的是 Yii 框架本身的安装 目录，而 @web 表示的是当前运行应用的根 URL。

**F、对象的配置**

[Object](http://www.yiichina.com/doc/api/2.0/yii-base-object) 类引入了一种统一对象配置的方法。 所有 [Object](http://www.yiichina.com/doc/api/2.0/yii-base-object) 的子类都应该用以下方法声明 它的构造方法（如果需要的话），以正确配置它自身：

class MyClass extends \yii\base\Object

{

public function \_\_construct($param1, $param2, $config = [])

{

// ... 配置生效前的初始化过程

parent::\_\_construct($config);

}

public function init()

{

parent::init();

// ... 配置生效后的初始化过程

}

}

**G、事件（Event）**

在 Yii 2 中，你可以使用任意的事件名， 同时通过调用 [trigger()](http://www.yiichina.com/doc/api/2.0/yii-base-component#trigger%28%29-detail) 方法来触发相关事件：

$event = new \yii\base\Event;

$component->trigger($eventName, $event);

要给事件附加一个事件事件处理器，需要使用 [on()](http://www.yiichina.com/doc/api/2.0/yii-base-component#on%28%29-detail) 方法：

$component->on($eventName, $handler);

// 解除事件处理器，使用 off 方法：

// $component->off($eventName, $handler);

**H、模型(Model)**

Yii 2.0 使用 [yii\base\Model](http://www.yiichina.com/doc/api/2.0/yii-base-model) 作为模型基类，现在要创建表单模型类，可以通过继承 [yii\base\Model](http://www.yiichina.com/doc/api/2.0/yii-base-model) 类来实现。

**I、路由**

当入口脚本在调用 yii\web\Application::run() 方法时，它进行的第一个操作就是解析输入的请求，然后实例化对应的控制器操作处理这个请求。 该过程就被称为引导路由（routing）。

**J、动作过滤器（Action Filters）**

动作的过滤现在通过行为（behavior）来实现。要定义一个新的，自定义的过滤器， 请继承 [yii\base\ActionFilter](http://www.yiichina.com/doc/api/2.0/yii-base-actionfilter) 类。要使用一个过滤器，需要把过滤器类作为一个 behavior 绑定到控制器上。 例如，要使用 [yii\filters\AccessControl](http://www.yiichina.com/doc/api/2.0/yii-filters-accesscontrol) 过滤器，你需要在控制器内添加如下代码：

public function behaviors()

{

return [

'access' => [

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

'rules' => [

['allow' => true, 'actions' => ['admin'], 'roles'=>['@']],

],

],

];

}

**K、表单**

Yii 2.0 引进了**表单栏（field）**的概念，用来创建一个基于[yii\widgets\ActiveForm](http://www.yiichina.com/doc/api/2.0/yii-widgets-activeform) 的表单。 一个表单栏是一个由标签、输入框、错误消息（可能还有提示文字）组成的容器， 被表示为一个 [ActiveField](http://www.yiichina.com/doc/api/2.0/yii-widgets-activefield) 对象。 使用表单栏建立表单的过程比以前更整洁利落：

<?php $form = yii\widgets\ActiveForm::begin(); ?>

<?= $form->field($model, 'username') ?>

<?= $form->field($model, 'password')->passwordInput() ?>

<div class="form-group">

<?= Html::submitButton('Login') ?>

</div>

<?php yii\widgets\ActiveForm::end(); ?>

\*/

**L、控制台应用（Console Application）**

控制台应用现在如普通的 Web 应用程序一样，由控制器组成，控制台的控制器继承自 [yii\console\Controller](http://www.yiichina.com/doc/api/2.0/yii-console-controller)，运行控制台命令使用 yii <route>，其中 <route> 代表控制器的路由（如 sitemap/index）。 额外的匿名参数传递到对应的控制器操作方法，而有名的参数根据 yii\console\Controller::options() 的声明来解析。

三、数据库说明

四、服务管理

服务管理的主要功能是对应用服务进行增、删、改、查；给“数据统计-应用数据”模块提供基础的服务档案。

代码结构如下：

控制器： \will\jsga.wiiqq.com\modules\app\controllers\\*

model类： \will\jsga.wiiqq.com\models\app\\*

ajax请求类：\will\jsga.wiiqq.com\modules\api\controllers\app\\*

views： \will\jsga.wiiqq.com\modules\app\views\\*

权限过滤filter配置文件：app.yml

五、用户中心 数据统计－用户数据

用户中心、用户统计模块为 \will\jsga.wiiqq.com\modules\usercenter；

接口在\will\jsga.wiiqq.com\modules\api\controllers\usercenter目录下；

用户中心展示的数据大多存在缓存中，由\will\jsga.wiiqq.com\models\task \SyncEsTask.php控制，具体实现在\code\will\jsga.wiiqq.com\models\redis中，

RedisService.php为用户中心、用户统计需要展示数据实现缓存的类。

注：如何查找当前页面所对应控制器以及接口类：

如当前为用户中心模块页面的地址为/usercenter/usercenter/overview

那么此页面对应的模块为usercenter

对应的控制器为UsercenterController

对应的方法为OverviewAction

此页面请求的接口为usercenter/usercenterapi/get\_user\_total

接口对应关系配置是在UsercenterapiController中，通过对应关系即可找到对应接口的实现类。

六、大屏展示

用户中心、用户统计模块为\will\jsga.wiiqq.com\modules\screen

接口在\will\jsga.wiiqq.com\modules\api\controllers\screen

用户中心展示的数据大多存在缓存中，由\will\jsga.wiiqq.com\models\task\ SyncEsTask.php控制，具体实现在\will\jsga.wiiqq.com\models\redis文件夹中，CoolService.php为大屏展示数据缓存实现类。

七、数据统计－应用数据

注意：1、yml具体的配置参考（<http://doc.wiiqq.com/wau/>）

2、左上角切换城市后所有数据均相应切换（参数switch\_city\_id）

3、echart 的具体配置参考 http://echarts.baidu.com/index.html

服务数据监测

**总览**

控制器： \will\jsga.wiiqq.com\modules\applies\controllers\AppliesController.php

ajax请求类：\will\jsga.wiiqq.com\modules\applies\controllers\AppliesapiController.php

views： \will\jsga.wiiqq.com\modules\applies\views\applies\\*

权限过滤： \will\jsga.wiiqq.com\config\filters\applies.yml

**服务趋势**

控制器： \will\jsga.wiiqq.com\modules\applies\controllers\AppliesController.php

ajax请求类：\will\jsga.wiiqq.com\modules\applies\controllers\AppliesapiController.php

views： \will\jsga.wiiqq.com\modules\applies\views\applies\ visit.yml

权限过滤： \will\jsga.wiiqq.com\config\filters\applies.yml

**服务对比**

控制器：

\will \jsga.wiiqq.com\modules\servicereport\controllers\ ServicereportController.php

ajax请求类：

\will \jsga.wiiqq.com\modules\servicereport\controllers\ ServicereportapiController.php

views：

\will\jsga.wiiqq.com\modules\servicereport\views\servicereport\\*

权限过滤：

\will\jsga.wiiqq.com\config\filters\ servicereport.yml

**办理日志**

控制器：

\will\jsga.wiiqq.com\modules\business\controllers\ BusinessController.php

ajax请求类：

\will\jsga.wiiqq.com\modules\business\controllers\ BusinessapiController.php

views：\will\jsga.wiiqq.com\modules\business\views\business\ log.yml

权限过滤：\will\jsga.wiiqq.com\config\filters\ business.yml

常用数据监测

**总览**

控制器：

\will\jsga.wiiqq.com\modules\ servicereport \controllers\ ServicereportController.php

ajax请求类：

\will\jsga.wiiqq.com\modules\ servicereport \controllers\ ServicereportapiController.php

views：

\will\jsga.wiiqq.com\modules\ servicereport \views\ servicereport \ common.yml

权限过滤：

\will\jsga.wiiqq.com\config\filters\ servicereport.yml

**自助移车**

控制器：

\will\jsga.wiiqq.com\modules\ servicereport \controllers\ ServicereportController.php

ajax请求类：

\will\jsga.wiiqq.com\modules\ servicereport \controllers\ ServicereportapiController.php

views：

\will\jsga.wiiqq.com\modules\ servicereport \views\ servicereport \ removal.yml

权限过滤：

\will\jsga.wiiqq.com\config\filters\ servicereport.yml

全流程应用

**总览**

控制器：

\will\jsga.wiiqq.com\modules\wholeprocess\controllers\ WholeprocessController.php

ajax请求类：

\will\jsga.wiiqq.com\modules\wholeprocess\controllers \ WholeprocessapiController.php

views：

\will\jsga.wiiqq.com\modules\ wholeprocess \views\ wholeprocess \ overview.yml

权限过滤：

\will\jsga.wiiqq.com\config\filters\ wholeprocess.yml

**办理日志**

控制器：

\will\jsga.wiiqq.com\modules\wholeprocess\controllers\ WholeprocessController.php

ajax请求类：

\will\jsga.wiiqq.com\modules\wholeprocess\controllers \ WholeprocessapiController.php

views：

\will\jsga.wiiqq.com\modules\ wholeprocess \views\ wholeprocess \ log.yml

权限过滤：

\will\jsga.wiiqq.com\config\filters\ wholeprocess.yml

八、定时任务说明

九、接口说明

接口返回说明请参考：江苏公安微警务综合服务管理平台接口文档v2.20.docx

内网接口

a、用户数据上报接口:

\will\jsga.wiiqq.com\modules\api\controllers\usercenter\SaveAction.php

由于历史数据状况比较多，此类的handlUser方法中，加入了一些数据处理的逻辑，在\app\models\usercenter\UserCenterMongodb::saveFieldsLogicHandle这个方法中进行当前数据的校验、处理

当保存用户到mongo数据表中后，会同步到es中，调用如下方法：\app\models\es\UserCenterEs::syncByMongoId

b、业务数据上报接口：\app\modules\api\controllers\business\SaveAction

上报逻辑处理：

\app\models\business\BusinessMongodb::saveFieldsLogicHandle

需要注意的是，保存业务日志的时候，会更新之前的同一条业务数据状态（业务id、app\_id分别相同），并且将此条记录成最新的一条，所有的统计查询都查询最新的业务日志（isnewest=1）

当保存业务日志到mongo数据表中后，会同步到es中，调用如下方法：\app\models\es\BusinessEs::syncByMongoId

c、访问pv上报：

\app\modules\api\controllers\business\VisitAction::runAction

d、警民互动上报：

\app\modules\api\controllers\interactive\SaveAction

当保存警民互动到mongo数据表中后，会同步到es中，调用如下方法：\app\models\es\InteractiveEs::syncByMongoId

外网接口

1、授权接口

静默授权：

\app\modules\api\controllers\AuthController::actionIndex

主动授权：

\app\modules\api\controllers\AuthController::actionAuth

说明：当外网授权完成之后，会将此用户的省厅openid、app\_id等保存到外网的mongo，参考\app\modules\api\controllers\AuthController::saveUser

这些数据，由定时任务调用内网的用户上报接口进行上报，参考：

\app\models\task\SendAuthorizationDataTask

2、警民互动上报接口

\app\modules\api\controllers\interactive\SaveInteractiveAction

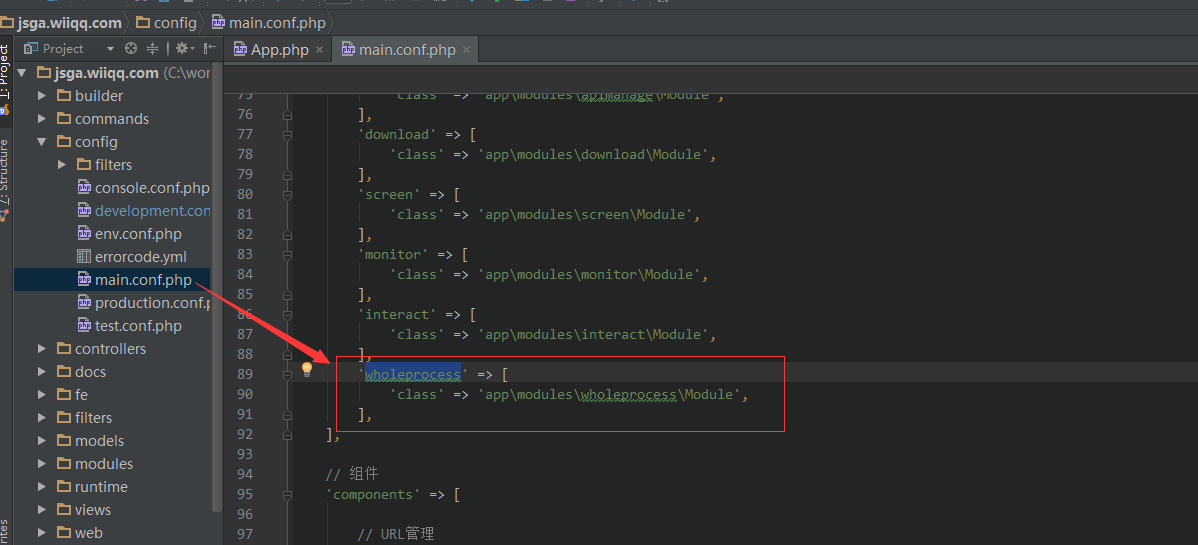
说明：当外网接受警民互动数据保存到mongo后，这些数据由定时任务上报到内网，参考

\app\models\task\InteractiveDataTask

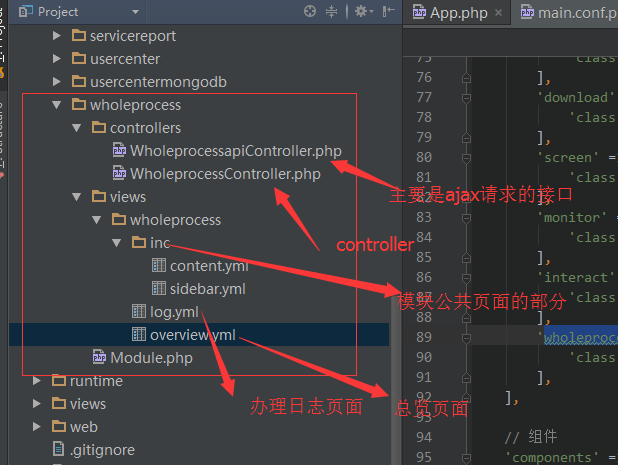
十、开发demo

**全流程应用开发流程（wholeprocess）**

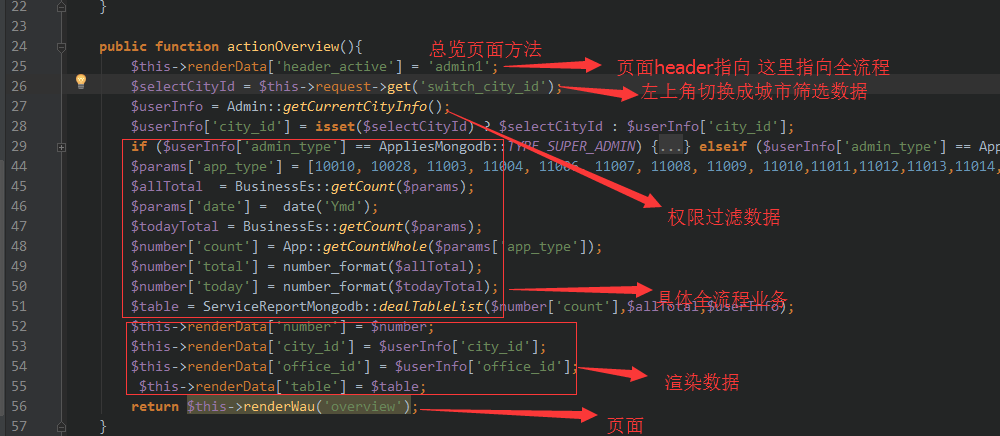
* + - 1. 建立模块 在modules中建立模块wholeprcess（modules/wholeprocess）
      2. 在配置文件中加入模块 config/main.conf.php



* + - 1. 在config/filters中加入filter （config/filters/wholeprocess.yml）
      2. 文件结构



* + - 1. 具体逻辑流程



* + - 1. yml具体的配置参考（<http://doc.wiiqq.com/wau/>）

wau是一套基于ACE封装的一套基于配置快速生成页面的管理后台系统，旨在帮助前端开发脱离管理系统开发的一套自动化后台系统，目前主要包括列表页、表单页、详情页、APP卡片页、外壳页，其它个性页面暂不做支持。