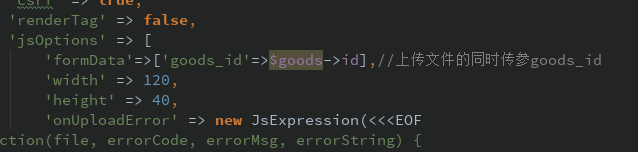
主要完成商品相册的添加、删除，管理员的增删改查和登陆注销。

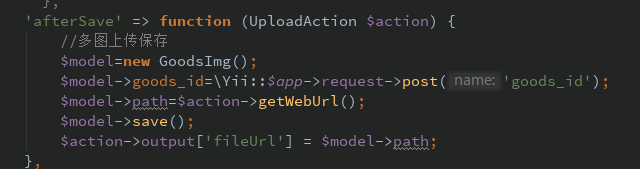
1、商品相册

使用了Uploadify插件，支持多图上传。主要代码：

将商品id传给控制器



保存数据到mysql



2、管理员

管理员的增删改查和其余表的差别不大

3、登陆注销

新建一个loginform.php（需要继承认证接口 yii\web\IdentityInterface） 主要用于验证账号密码是否和数据库里的一致

控制器将接受到的数据传到模型验证，正确则跳到首页，错误则跳转到登陆页面

登陆验证：Yii::$app->user->login();

注销：Yii::$app->user->logout();

基于RBAC的角色权限管理

1、在common\config\main.php 里添加：

'authManager' => [  
 'class' => 'yii\rbac\DbManager',  
],

2、项目主目录下打开命令窗口，输入：yii migrate --migrationPath=@yii/rbac/migrations

创建权限管理表

3、

a 设置权限

控制器：

//添加权限  
**public function** actionAddPermission(){  
 $model=**new** PermissionForm();  
 **if**($model->load(\Yii::*$app*->request->post()) && $model->validate()){  
 **if**($model->addPermission()){  
 \Yii::*$app*->session->setFlash('权限添加成功！');  
 **return** $this->redirect('permission-index');  
 }  
 }  
 **return** $this->render('add-permission',['model'=>$model]);  
}

模型：

//添加权限方法  
**public function** addPermission(){  
 $authManager=\Yii::*$app*->authManager;  
 **if**($authManager->getPermission($this->name)){  
 $this->addError('name','权限已存在');  
 }**else**{  
 $permission=$authManager->createPermission($this->name);  
 $permission->description=$this->description;  
 **return** $authManager->add($permission);  
 }  
 **return false**;  
}

修改：

通过name得到权限

$permission=\Yii::*$app*->authManager->getPermission($name);

修改

//判断修改后的权限是否存在 如果修改了名字并且修改后的名字已存在则返回false  
**if**($name!=$this->name && $authManager->getPermission($this->name)){  
 $this->addError('name','权限已存在');  
}**else**{  
 $permission->name=$this->name;  
 $permission->description=$this->description;  
 **return** $authManager->update($name,$permission);  
}

b 设置角色

控制器：

//修改角色  
**public function** actionEditRole($name){  
 $role=\Yii::*$app*->authManager->getRole($name);  
 **if**($role==**null**){  
 **throw new** NotFoundHttpException('角色不存在！');  
 }  
 $model=**new** RoleForm();  
 //已有的权限默认选中 当$model->permission中有值的时候便自动选择  
 $permission=\Yii::*$app*->authManager->getPermissionsByRole($name);  
 **foreach** ($permission **as** $permissionName){  
 $model->permissions[]=$permissionName->name;  
 }  
  
 $model->loadDate($role);  
 **if**($model->load(\Yii::*$app*->request->post()) && $model->validate()){  
 **if**($model->updateRole($name)){  
 \Yii::*$app*->session->setFlash('修改角色成功');  
 **return** $this->redirect('role-index');  
 }  
 }  
 **return** $this->render('add-role',['model'=>$model]);  
}

模型：

**public function** loadDate(Role $role){  
// var\_dump($role);exit;  
 $this->name=$role->name;  
 $this->description=$role->description;  
 //通过角色获取到权限  
 $permissions=\Yii::*$app*->authManager->getPermissionsByRole($role->name);  
 **foreach** ($permissions **as** $permission){  
 $this->permissions[]=$permission;  
 }  
 }  
 **public function** updateRole($name){  
 $authManager=\Yii::*$app*->authManager;  
 $role = $authManager->getRole($name);  
 $role->name=$this->name;  
 $role->description=$this->description;  
 //如果角色名修改且修改后的名字已存在，则给出错误提示  
 **if**($name!=$this->name && $authManager->getRole($this->name)){  
 $this->addError('name','角色名重复');  
 }**else**{  
 //如果成功修改了角色权限 去掉该角色所有的权限然后再重新添加权限  
 **if**($authManager->update($name,$role)){  
 $authManager->removeChildren($role);  
 //添加权限  
 **if**($this->permissions){  
 **foreach** ($this->permissions **as** $permissionName){  
 $permission=$authManager->getPermission($permissionName);  
 **if**($permission){  
 $authManager->addChild($role,$permission);  
 }  
 }  
 }  
 }  
 **return true**;  
 }  
 **return false**;  
 }

c 绑定用户和角色

控制器

//将角色和用户关联  
**public function** actionRole($id){  
 $model = AdminUser::*findOne*(['id'=>$id]);  
 $username=$model->username;  
 //已有的角色默认选中 当$model->role中有role值的时候便自动选中  
 $roles =\Yii::*$app*->authManager->getRolesByUser($id);  
 **foreach** ($roles **as** $role){  
 $model->role[] = $role->name;  
 }  
 **if** ($model->load(\Yii::*$app*->request->post())) {  
 **if**($model->updateRole($model,$id)){  
 \Yii::*$app*->session->setFlash('修改权限成功');  
 **return** $this->redirect('index');  
 }  
 }  
 **return** $this->render('role', ['model' => $model,'username'=>$username]);  
}

模型：

**public function** updateRole($model,$id){  
 $authManager = \Yii::*$app*->authManager;  
 //找到该用户的所有权限  
 $authManager->getRolesByUser($id);  
 //如果用户有角色则执行下面的步骤  
 **if**($model->role){  
 //去掉该用户所有的角色然后再重新添加  
 $res=$authManager->revokeAll($id);  
 **foreach** ($model->role **as** $roleName){  
 $role= $authManager->getRole($roleName);  
 **if**($role){  
 $authManager->assign($role,$id);  
 }  
 }  
 }  
 **return true**;  
}

4、验证权限

在控制器需要权限的方法下加入代码：

**if** (!\Yii::*$app*->user->can('article\_add')) {  
 **throw new** ForbiddenHttpException('对不起，权限不足');  
}

菜单管理

新建MenuWidget.php菜单管理工具，根据登陆用户的权限决定展示那些菜单

$menuItems=[];  
//获取一级菜单  
$menus=Menu::*findAll*(['parent\_id'=>0]);  
//获取所有一级菜单  
$menus = Menu::*findAll*(['parent\_id'=>0]);  
**foreach** ($menus **as** $menu){  
 $item = ['label'=>$menu->label,'items'=>[]];  
 **foreach** ($menu->children **as** $child){  
 //根据用户权限判断，该菜单是否显示  
 **if**(Yii::*$app*->user->can($child->url)){  
 $item['items'][] = ['label'=>$child->label,'url'=>[$child->url]];  
 }  
  
 }  
 //如果该一级菜单没有子菜单，就不显示  
 **if**(!**empty**($item['items'])){  
 $menuItems[] = $item;  
 }

在components里新建RbacFilter.php权限管理

**class** RbacFilter **extends** ActionFilter{  
 **public function** beforeAction($action)  
 {  
 $user = \Yii::*$app*->user;  
 **if**(!$user->can($action->uniqueId)){  
 //如果用户未登录，引导用户先登录  
 **if**($user->isGuest){  
 **return** $action->controller->redirect($user->loginUrl);  
 }  
 **throw new** HttpException(403,'对不起,权限不足!');  
 **return false**;  
 }  
 **return parent**::*beforeAction*($action); // *TODO: Change the autogenerated stub* }  
}