```php
<?php
namespace app\api\controller;
use app\api\service\Card as CardService;
use app\base\AdminController;
use app\p\logic\MealOrder as MealOrderServer;
use think\Exception;
use think\App;
use think\response\Json;
use app\api\service\Consume as ConsumeService;
class Consume extends AdminController
{
    public function dishes(): Json
    {
        $page_size = $this->request->post('page_size', 20);
        $page_size = intval($page_size);
        $category = $this->request->post('category', 0);
        $category = in_array($category, [0,1,2,3,4,5,6]) ? $category : 0;
        $keyword = $this->request->post('keyword', '');
        $keyword = trim($keyword);
        $ConsumeService = new ConsumeService;
        $data = $ConsumeService->dishes($page_size, $category, $keyword);
        return json(['code' => 0, 'message' => '查询成功', 'data' => $data]);
    }
    public function dishesRecognizeImgUpload(): Json
    {
        try {
            $dishesUuid = trim((string)$this->request->post('dishes_uuid', ''));
            $feature = trim((string)$this->request->post('feature', ''));
            $equipmentCode = trim((string)$this->request->post('equipment_code', ''));
            $file = $this->request->file('file');
            if ($dishesUuid === '' || $feature === '' || $equipmentCode === '' || empty($file)) {
                throw new Exception('缺少参数');
            }
            $consumeService = new ConsumeService;
            $data = $consumeService->uploadDishesRecognizeImg($dishesUuid, $file, $feature, $equipmentCode);
        } catch (\Throwable $e) {
            return json(['code' => 1, 'message' => $e->getMessage(), 'data' => []]);
        }
        return json(['code' => 0, 'message' => '上传成功', 'data' => $data]);
    }
    public function dishesRecognizeImgList(): Json
    {
        try {
            $page = (int)$this->request->post('page', 1);
            $pageSize = (int)$this->request->post('page_size', 20);
            $page = $page > 0 ? $page : 1;
            $pageSize = $pageSize > 0 ? $pageSize : 20;
            $consumeService = new ConsumeService;
            $data = $consumeService->getDishesRecognizeImgList($page, $pageSize);
        } catch (\Throwable $e) {
```

```
            return json(['code' => 1, 'message' => $e->getMessage(), 'data' => []]);
        }
        return json(['code' => 0, 'message' => '查询成功', 'data' => $data]);
    }
    public function dishesRecognizeImgDelete(): Json
    {
        try {
            $ids = $this->request->post('ids/a', []);
            if (empty($ids)) {
                throw new Exception('缺少参数');
            }
            $consumeService = new ConsumeService;
            $data = $consumeService->deleteDishesRecognizeImg($ids);
        } catch (\Throwable $e) {
            return json(['code' => 1, 'message' => $e->getMessage(), 'data' => []]);
        }
        return json(['code' => 0, 'message' => '删除成功', 'data' => $data]);
    }
}
    public function dishes($page_size, $category = 0, $keyword = '')
    {
        $where = [
            'is_del' => 1,
        ];
        if (!empty($category)) {
            $where['category'] = $category;
        }
        if (!empty($keyword)) {
            $where['name'] = ['like', "%{$keyword}%"];
        }
        $field = ['id','uuid','name','price','file_path','unit'];
        $DishesModel = new DishesModel;
        $data = $DishesModel->where($where)
            ->field($field)
            ->order(['id' => 'asc'])
            ->paginate($page_size)
            ->toArray();
        if (!empty($data['data'])) {
            $unit = ConfigEnumModel::getEnum(EnumType::ENUM3);
            $unit = array_column($unit, 'enum_name', 'enum_key');
            foreach ($data['data'] as $key => $item) {
                $data['data'][$key]['unit'] = $unit[$item['unit']] ?? '';
            }
        }
        return $data;
    }
    public function uploadDishesRecognizeImg(string $dishesUuid, $file, string $feature, string $equipmentCode): array
    {
        $dishesInfo = (new DishesModel())
            ->where('is_del', 1)
```

```
        ->where('uuid', $dishesUuid)
        ->field(['uuid', 'name'])
        ->find();
    if (empty($dishesInfo)) {
        throwError('菜品不存在');
    }
    $modulePath = $this->fileModel->getFilePath('dishesRecognizeImg');
    $upload = $this->fileModel->upload($file, $modulePath, [], guid(), true, false, [200, 200], false);
    if (($upload['code'] ?? 1) !== 0 || empty($upload['data'])) {
        throwError($upload['message'] ?? '图片上传失败');
    }
    $data = [
        'dishes_uuid' => $dishesUuid,
        'img_url' => (string)$upload['data'],
        'feature' => $feature,
        'equipment_code' => $equipmentCode,
        'is_delete' => 0,
        'create_time' => date('Y-m-d H:i:s'),
        'update_time' => date('Y-m-d H:i:s'),
    ];
    $id = $this->dishesRecognizeImgModel->add($data);
    if (!$id) {
        throwError('菜品识别图片保存失败');
    }
    $data['id'] = (int)$id;
    $data['img_url'] = getImgFullPath($data['img_url']);
    return $data;
}
public function getDishesRecognizeImgList(int $page, int $pageSize): array
{
    $data = $this->dishesRecognizeImgModel
        ->alias('d')
        ->join('dishes c', 'c.uuid = d.dishes_uuid')
        ->where([
            'd.is_delete' => 0,
            'c.is_del' => 1,
        ])
        ->field(['d.id', 'd.dishes_uuid', 'd.img_url', 'd.feature'])
        ->order(['id' => 'desc'])
        ->page($page, $pageSize)
        ->select()
        ->toArray();
    foreach ($data as $key => $item) {
        $data[$key]['img_url'] = getImgFullPath($item['img_url']);
    }
    $total = $this->dishesRecognizeImgModel
        ->alias('d')
        ->join('dishes c', 'c.uuid = d.dishes_uuid')
        ->where([
            'd.is_delete' => 0,
```

```
            'c.is_del' => 1,
        ])
        ->count();
    return [
        'data' => $data,
        'total' => (int)$total,
        'page' => $page,
        'page_size' => $pageSize,
    ];
}
public function deleteDishesRecognizeImg(array $ids): array
{
    $ids = array_values(array_unique(array_filter(array_map('intval', $ids))));
    if (empty($ids)) {
        throwError('缺少参数');
    }
    $rows = $this->dishesRecognizeImgModel
        ->where('is_delete', 0)
        ->whereIn('id', $ids)
        ->field(['id'])
        ->select()
        ->toArray();
    if (empty($rows)) {
        return [
            'ids' => [],
            'delete_count' => 0,
        ];
    }
    $deleteIds = array_column($rows, 'id');
    $affected = $this->dishesRecognizeImgModel
        ->whereIn('id', $deleteIds)
        ->update([
            'is_delete' => 1,
            'update_time' => date('Y-m-d H:i:s'),
        ]);
    if ($affected === false) {
        throwError('删除失败');
    }
    return [
        'ids' => $deleteIds,
        'delete_count' => (int)$affected,
    ];
}
public function dishesImg() {
    try {
        $uuid = $this->request->post('uuid');
        if(!$uuid)
            throw new Exception('参数错误');
        $res = $this->BillService->dishesImg($uuid);
        if($res['code'])
```

```
            throw new Exception($res['message']);
        }catch(Exception $e){
            return json(['code' => 1, 'message' => $e->getMessage(), 'data'=>[]]);
        }
        return json(['code' => 0, 'message' => "获取成功", 'data' => $res['data']]);
    }
    public function dishesRecognizeImgList() {
        try {
            $page = (int)$this->request->post('page', 1);
            $page_size = (int)$this->request->post('page_size', 20);
            $keyword = trim((string)$this->request->post('keyword', ''));
            $page = $page > 0 ? $page : 1;
            $page_size = $page_size > 0 ? $page_size : 20;
            $res = $this->BillService->dishesRecognizeImgList($page, $page_size, $keyword);
            if($res['code'])
                throw new Exception($res['message']);
        }catch(Exception $e){
            return json(['code' => 1, 'message' => $e->getMessage(), 'data'=>[]]);
        }
        return json(['code' => 0, 'message' => "获取成功", 'data' => $res['data']]);
    }
    public function dishesRecognizeImgDelete() {
        try {
            $ids = $this->request->post('ids/a', []);
            if(empty($ids))
                throw new Exception('参数错误');
            $res = $this->BillService->dishesRecognizeImgDelete($ids);
            if($res['code'])
                throw new Exception($res['message']);
        }catch(Exception $e){
            return json(['code' => 1, 'message' => $e->getMessage(), 'data'=>[]]);
        }
        return json(['code' => 0, 'message' => "删除成功", 'data' => $res['data']]);
    }
    public function dishesImg($uuid)
    {
        $res = Db::name('dishes_hardware_img')
            ->where(['dishes_uuid' => $uuid])
            ->field(['head_img'])
            ->select();
        return array('code' => 0, 'message' => '获取成功', 'data' => $res);
    }
    public function dishesRecognizeImgList($page, $page_size, $keyword = '')
    {
        try {
            $where = [
                'd.is_delete' => 0,
                'c.is_del' => 1,
            ];
            $query = Db::name('dishes_recognize_img')
```

```
        ->alias('d')
        ->join('dishes c', 'c.uuid = d.dishes_uuid')
        ->where($where);
    if ($keyword !== '') {
        $query->where('c.name', 'like', '%' . $keyword . '%');
    }
    $total = count((clone $query)
        ->group('d.dishes_uuid')
        ->column('d.dishes_uuid'));
    $dishesRows = (clone $query)
        ->field(['d.dishes_uuid', 'c.name as dishes_name', 'c.price', 'count(d.id) as image_count', 'max(d.id) as latest_id'])
        ->group('d.dishes_uuid')
        ->order(['latest_id' => 'desc'])
        ->page($page, $page_size)
        ->select();
    if ($dishesRows instanceof \think\Collection) {
        $dishesRows = $dishesRows->toArray();
    } elseif (!is_array($dishesRows)) {
        $dishesRows = (array)$dishesRows;
    }
    if (empty($dishesRows)) {
        return array('code' => 0, 'message' => '获取成功', 'data' => [
            'data' => [],
            'total' => 0,
            'page' => (int)$page,
            'page_size' => (int)$page_size,
        ]);
    }
    $dishesUuids = array_column($dishesRows, 'dishes_uuid');
    $imgRows = $this->dishesRecognizeImgModel
        ->where('is_delete', 0)
        ->whereIn('dishes_uuid', $dishesUuids)
        ->field(['id', 'dishes_uuid', 'img_url', 'feature', 'equipment_code', 'create_time'])
        ->order(['id' => 'desc'])
        ->select();
    if ($imgRows instanceof \think\Collection) {
        $imgRows = $imgRows->toArray();
    } elseif (!is_array($imgRows)) {
        $imgRows = (array)$imgRows;
    }
    $imgMap = [];
    foreach ($imgRows as $imgRow) {
        $imgRow['img_url'] = getImgFullPath($imgRow['img_url']);
        $imgMap[$imgRow['dishes_uuid']][] = $imgRow;
    }
    foreach ($dishesRows as $key => $row) {
        $dishesRows[$key]['images'] = $imgMap[$row['dishes_uuid']] ?? [];
        $dishesRows[$key]['price'] = isset($row['price']) ? (string)$row['price'] : '0.00';
        $dishesRows[$key]['image_count'] = (int)$row['image_count'];
        unset($dishesRows[$key]['latest_id']);
```

```
            }
            return array('code' => 0, 'message' => '获取成功', 'data' => [
                'data' => $dishesRows,
                'total' => (int)$total,
                'page' => (int)$page,
                'page_size' => (int)$page_size,
            ]);
        } catch (Exception $e) {
            return array('code' => 1, 'message' => $e->getMessage(), 'data' => []);
        }
    }
    public function dishesRecognizeImgDelete($ids)
    {
        try {
            $consumeService = new ConsumeService();
            $data = $consumeService->deleteDishesRecognizeImg((array)$ids);
            return array('code' => 0, 'message' => '删除成功', 'data' => $data);
        } catch (Exception $e) {
            return array('code' => 1, 'message' => $e->getMessage(), 'data' => []);
        }
<?php
namespace app\p\controller;
use app\base\AdminController;
use think\Exception;
use app\validate\Equipment as EquipmentValidate;
use app\p\logic\Equipment as EquipmentLogic;
use app\common\enum\EquipmentType as EquipmentTypeEnum;
class Equipment extends AdminController {
    protected $beforeActionList = ['checkLogin', 'checkFunctionPermission'=>['only'=>'add,update,del']];
    public $logic;
    public $validate;
    public function __construct(\think\Request $request = null) {
        parent::__construct($request);
        $this->logic = new EquipmentLogic();
        $this->validate = new EquipmentValidate();
        $this->currentUser = $this->getCurrentUser();
    }
    private function applyRestaurantPermissionWhere(array &$where, $requestedRestaurantId = ''): void
    {
        $permittedIds = $this->resolvePermittedRestaurantIds($requestedRestaurantId);
        if ($permittedIds !== null) {
            $where['restaurant_id'] = empty($permittedIds) ? 0 : ['in', $permittedIds];
        }
    }
    private function hasEquipmentRestaurantPermission($equipmentId): bool
    {
        $restaurantId = \think\Db::name('equipment')
            ->where('id', (int)$equipmentId)
            ->where('del_flag', 0)
            ->value('restaurant_id');
```

```
    return !empty($restaurantId) && $this->hasRestaurantPermission($restaurantId);
}
private function normalizeRestaurantId($restaurantId): int
{
    if (is_array($restaurantId)) {
        $restaurantId = end($restaurantId);
    }
    return (int)($restaurantId ?: 1);
}
public function lists() {
    $page = $this->request->post('page',1);
    $pageSize = $this->request->post("pageSize", \think\Config::get("pagesize"));
    $type = $this->request->post("type");
    $where = ['del_flag'=>0];
    if(!empty($type)) {
        $where['type'] = $type;
    }
    $this->applyRestaurantPermissionWhere($where, $this->request->post('restaurant_id', ''));
    $orderBy = ['id'=>'desc'];
    $res['rows'] = $this->logic->lists($where, $page, $pageSize, $orderBy);
    $res['total'] = $this->logic->totalCount($where);
    return json(['code' => 0, 'message' => "", 'data' => $res]);
}
public function add() {
    $params = $this->request->post();
    try {
        $params['restaurant_id'] = $this->normalizeRestaurantId($params['restaurant_id'] ?? 1);
        if (!$this->hasRestaurantPermission($params['restaurant_id'])) {
        }
        if(!$this->validate->scene('add')->check($params)) {
            throw new Exception($this->validate->getError());
        }
        $res = $this->logic->add($params, $this->currentUser);
        if($res['code']) {
            throw new Exception($res['message']);
        }
    } catch (Exception $ex) {
        return json(['code' => 1,'message' => $ex->getMessage(), 'data'=>[]]);
    }
    return json(['code' => 0,'message' => '添加成功', 'data'=>[]]);
}
public function read() {
    $id = $this->request->post("id");
    if (!$this->hasEquipmentRestaurantPermission($id)) {
    }
    $res = $this->logic->read($id);
    return json(['code' => 0,'message' => '', 'data'=>$res]);
}
public function update() {
    $params = $this->request->post();
```

```
        try {
            if (!$this->hasEquipmentRestaurantPermission($params['id'] ?? 0)) {
            }
            if (isset($params['restaurant_id'])) {
                $params['restaurant_id'] = $this->normalizeRestaurantId($params['restaurant_id']);
                if (!$this->hasRestaurantPermission($params['restaurant_id'])) {
                }
            }
            if(!$this->validate->scene('update')->check($params)) {
                throw new Exception($this->validate->getError());
            }
            $res = $this->logic->update($params, $this->currentUser);
            if($res['code']) {
                throw new Exception($res['message']);
            }
        } catch (Exception $ex) {
            return json(['code' => 1,'message' => $ex->getMessage(), 'data'=>[]]);
        }
        return json(['code' => 0,'message' => '修改成功', 'data'=>[]]);
    }
    public function del() {
        $ids = trim($this->request->post("ids"));
        try {
            if(empty($ids)) {
                throw new Exception("参数错误");
            }
            foreach (array_filter(explode(',', $ids)) as $id) {
                if (!$this->hasEquipmentRestaurantPermission($id)) {
                }
            }
            $res = $this->logic->del($ids, $this->currentUser);
            if($res['code']) {
                throw new Exception($res['message']);
            }
        } catch (Exception $ex) {
            return json(['code' => 1,'message' => $ex->getMessage(), 'data'=>[]]);
        }
        return json(['code' => 0,'message' => '删除成功', 'data'=>[]]);
    }
    public function genCode() {
        $id = $this->request->post("id");
        if (!$this->hasEquipmentRestaurantPermission($id)) {
        }
        $res = $this->logic->genCode($id, $this->currentUser);
        return json($res);
    }
        $id = $this->request->post("id");
        if (!$this->hasEquipmentRestaurantPermission($id)) {
        }
        return json($res);
```

```
}
public function getCode(){
    $type = $this->request->post('type');
    if (empty($type)){
        return json(['code' => 0, 'message' => "参数错误", 'data' => []]);
    }
    $res = $this->logic->getCode($type);
    return json(['code' => 0, 'message' => "获取成功", 'data' => $res['code']]);
}
public function addVersion(){
    $params = $this->request->post();
    try {
        if(!$this->validate->scene('add_version')->check($params)) {
            throw new Exception($this->validate->getError());
        }
        $res = $this->logic->addVersion($params, $this->currentUser);
        if($res['code']) {
            throw new Exception($res['message']);
        }
    } catch (Exception $ex) {
        return json(['code' => 1,'message' => $ex->getMessage(), 'data'=>[]]);
    }
    return json(['code' => 0,'message' => '添加成功', 'data'=>[]]);
}
public function versionList(){
    $page = $this->request->post('page',1);
    $pageSize = $this->request->post("pageSize", \think\Config::get("pagesize"));
    $type = $this->request->post("type");
    $where = [];
    if(!empty($type)) {
        $where['type'] = $type;
    }
    $orderBy = ['create_time'=>'desc'];
    $res = $this->logic->versionList($where, $page, $pageSize, $orderBy);
    return json(['code' => 0, 'message' => "", 'data' => $res]);
}
public function versionUpdate(){
    $params = $this->request->post();
    try {
        $res = $this->logic->versionUpdate($params, $this->currentUser);
        if($res['code']) {
            throw new Exception($res['message']);
        }
    } catch (Exception $ex) {
        return json(['code' => 1,'message' => $ex->getMessage(), 'data'=>[]]);
    }
    return json(['code' => 0,'message' => '修改成功', 'data'=>[]]);
}
public function versionRead(){
    $uuid = $this->request->post("uuid");
```

```
    $res = $this->logic->versionRead($uuid);
    return json(['code' => 0,'message' => '', 'data'=>$res]);
}
public function editEquipmentByCode()
{
    $code = trim($this->request->post("code"));
    $restaurant_id = trim($this->request->post("restaurant_id"));
    try {
        if(empty($code)||empty($restaurant_id)) {
            throw new Exception("参数错误");
        }
        if (!$this->hasRestaurantPermission($restaurant_id)) {
        }
        $res = $this->logic->editEquipmentByCode($code, $restaurant_id);
        if($res['code']) {
            throw new Exception($res['message']);
        }
    } catch (Exception $ex) {
        return json(['code' => 1,'message' => $ex->getMessage(), 'data'=>[]]);
    }
    return json(['code' => 0,'message' => '修改成功', 'data'=>[]]);
}
public function pushList() {
    $page = $this->request->post('page',1);
    $pageSize = $this->request->post("pageSize", \think\Config::get("pagesize"));
    $type = $this->request->post("type");
    $weighType  = EquipmentTypeEnum::weighType();
    $where = [
        'del_flag' => 0,
        'type' => ['in', $weighType]
    ];
    if (in_array($type, $weighType)) {
        $where['type'] = $type;
    }
    $this->applyRestaurantPermissionWhere($where, $this->request->post('restaurant_id', ''));
    $orderBy = ['id'=>'desc'];
    $res['rows'] = $this->logic->pushList($where, $page, $pageSize, $orderBy);
    $res['total'] = $this->logic->totalCount($where);
    return json(['code' => 0, 'message' => "查询成功", 'data' => $res]);
}
public function pushDishes() {
    $equipment_id = $this->request->post('equipment_id');
    $left_dishes_uuid = $this->request->post('left_dishes_uuid');
    $right_dishes_uuid = $this->request->post('right_dishes_uuid');
    if (!$this->hasEquipmentRestaurantPermission($equipment_id)) {
    }
    $res = $this->logic->pushDishes($equipment_id, $left_dishes_uuid, $right_dishes_uuid, $this->currentUser);
    return json($res);
}
public function plateList() {
```

```
    $page = $this->request->post('page',1);
    $pageSize = $this->request->post("pageSize", \think\Config::get("pagesize"));
    $where = [
        'del_flag' => 0,
        'type' => 8
    ];
    $this->applyRestaurantPermissionWhere($where, $this->request->post('restaurant_id', ''));
    $orderBy = ['id'=>'desc'];
    $field = ['id','code','name', 'type', 'update_time', 'update_by'];
    $res['rows'] = $this->logic->plateList($where, $page, $pageSize, $orderBy, $field);
    $res['total'] = $this->logic->totalCount($where);
    return json(['code' => 0, 'message' => "查询成功", 'data' => $res]);
}
public function plateDetail() {
    $equipment_id = $this->request->post('equipment_id');
    $menu_date = $this->request->post('menu_date');
    if (empty($equipment_id) || empty($menu_date)) {
        return json(['code' => 1, 'message' => '缺少参数', 'data' => []]);
    }
    if (!$this->hasEquipmentRestaurantPermission($equipment_id)) {
    }
    $data = $this->logic->plateDetail($equipment_id, $menu_date);
    return json(['code' => 0, 'message' => "查询成功", 'data' => $data]);
}
public function plateUpdate() {
    $equipment_id = $this->request->post('equipment_id');
    $menu_date = $this->request->post('menu_date');
    $morning_file_id = $this->request->post('morning_file_id', 0);
    $noon_file_id = $this->request->post('noon_file_id', 0);
    $night_file_id = $this->request->post('night_file_id', 0);
    $night_snack_file_id = $this->request->post('night_snack_file_id', 0);
    if (empty($equipment_id) || empty($menu_date)) {
        return json(['code' => 1, 'message' => '缺少参数', 'data' => []]);
    }
    if (!$this->hasEquipmentRestaurantPermission($equipment_id)) {
    }
    $params = [
        'equipment_id' => $equipment_id,
        'menu_date' => $menu_date,
        'morning_file_id' => $morning_file_id,
        'noon_file_id' => $noon_file_id,
        'night_file_id' => $night_file_id,
        'night_snack_file_id' => $night_snack_file_id,
    ];
    $res = $this->logic->plateUpdate($params, $this->currentUser);
    return json($res);
}
public function consumeList() {
    $consumeType  = EquipmentTypeEnum::consumeType();
    $where = [
```

```
            'del_flag' => 0,
            'type' => ['in', $consumeType]
        ];
        $this->applyRestaurantPermissionWhere($where, $this->request->post('restaurant_id', ''));
        $orderBy = [];
        $field = ['id','code','name'];
        $data = $this->logic->consumeList($where, $orderBy, $field);
        return json(['code' => 0, 'message' => "查询成功", 'data' => $data]);
    }
}
<?php
namespace app\p\logic;
use app\common\enum\EquipmentType as EquipmentTypeEnum;
use app\model\Dishes as DishesModel;
use app\common\model\DishesCate as DishesCateModel;
use app\model\EquipmentVersion;
use think\Db;
use think\Exception;
use app\model\Equipment as EquipmentModel;
use app\api\service\xpyun\Xpyun as XpyunService;
use app\common\model\EquipmentPushDishes as EquipmentPushDishesModel;
use app\model\MqMessageLog;
use app\model\MqMessage;
use app\common\model\UploadFile;
use app\common\model\PlateDishes;
use think\Log;
class Equipment {
    public $model;
    public $VersionModel;
    public $statusArr = ['1'=>'在线', '2'=>'离线'];
    public $typeArr = [];
    public function __construct() {
        $this->model = new EquipmentModel();
        $this->VersionModel = new EquipmentVersion();
        $this->typeArr = array_column(EquipmentTypeEnum::data(), 'name', 'value');
    }
    public function lists($where=[], $page=1, $pageSize=15, $orderBy=[], $field=['*']) {
        $equipments = $this->model->inquiry($where, $page, $pageSize, $orderBy, $field);
        if (!empty($equipments)) {
            $onlineStatus = [];
            if ($configHelper->isEnabled()) {
                $codes = array_filter(array_column($equipments, 'code'));
                if (!empty($codes)) {
                    try {
                        $onlineStatus = $statusService->queryOnlineStatus($codes);
                    } catch (\Throwable $throwable) {
                    }
                }
            }
            foreach($equipments as $key=>$equipment) {
```

```
            $equipments[$key]['type_cn'] = empty($this->typeArr[$equipment['type']]) ? '' : $this->typeArr[$equipment['type']];
            $equipments[$key]['status_cn'] = empty($this->statusArr[$equipment['status']]) ? '' : $this->statusArr[$equipment['status']];
            $onlineValue = null;
            if (!empty($equipment['code']) && array_key_exists($equipment['code'], $onlineStatus)) {
                $onlineValue = (int)$onlineStatus[$equipment['code']];
            }
        }
    }
    return $equipments;
}
public function totalCount($where=[]) {
    return $this->model->inquiryCount($where);
}
public function add($params, $currentUser) {
    $datetime = date("Y-m-d H:i:s");
    $operator = $currentUser['name'];
    try {
        Db::startTrans();
        $createdPrinterSn = null;
        $code_exist = $this->model->inquiryCount(['code'=>$params['code'],'del_flag'=>0]);
        if($code_exist) {
            throw new Exception("设备编号已存在");
        }
        $name_exist = $this->model->inquiryCount(['name'=>$params['name'],'del_flag'=>0]);
        if($name_exist) {
            throw new Exception("设备名称已存在");
        }
        $params['communicationkey'] = str_replace("-", "", guid());
        $params['create_time'] = $datetime;
        $params['create_by'] = $operator;
        $params['update_time'] = $datetime;
        $params['update_by'] = $operator;
        $equipmentId = $this->model->add($params);
        if(!$equipmentId) {
            throw new Exception("操作失败");
        }
        if (isset($params['type']) && (int)$params['type'] === 11) {
            $sn = isset($params['sn']) ? trim($params['sn']) : '';
            if ($sn === '') {
                throw new Exception('打印机 SN 不能为空');
            }
            $printerName = config('business_name') . '_' . $params['name'];
            $xpyun = new XpyunService();
            $xpRes = $xpyun->addPrinters([
                ['sn' => $sn, 'name' => $printerName]
            ]);
            if (!isset($xpRes['code']) || (int)$xpRes['code'] !== 0) {
                $msg = isset($xpRes['message']) ? (string)$xpRes['message'] : '添加打印机失败';
                throw new Exception($msg);
            }
```

```
            $data = isset($xpRes['data']) ? $xpRes['data'] : null;
            $successList = (is_array($data) && isset($data['success']) && is_array($data['success'])) ? $data['success'] : [];
            if (!in_array($sn, $successList, true)) {
                $failReason = '';
                if (is_array($data) && isset($data['failMsg']) && is_array($data['failMsg'])) {
                    foreach ($data['failMsg'] as $fm) {
                        if (is_string($fm) && strpos($fm, $sn . ':') === 0) {
                            $failReason = substr($fm, strlen($sn) + 1);
                            break;
                        }
                    }
                }
                $reasonMap = [
                    '1001' => '打印机编号和用户不匹配',
                    '1002' => '打印机未注册',
                    '1009' => '添加打印机时编号或名称不能为空',
                    '1010' => '打印机设备编号无效',
                    '1011' => '打印机已存在',
                    '1012' => '添加打印设备失败',
                ];
                $msg = '添加打印机失败';
                if ($failReason !== '') {
                    $msg = isset($reasonMap[$failReason]) ? $reasonMap[$failReason] : ('添加打印机失败（错误码 ' . $failReason . ' ）');
                }
                throw new Exception($msg);
            }
            $createdPrinterSn = $sn;
        }
        $new = $this->model->readById($equipmentId);
        Db::commit();
        return ['code' => 0,'message' => '操作成功', 'data'=>$new];
    } catch(Exception $ex) {
        try { Db::rollback(); } catch (\Throwable $t) {}
        try {
            if (!empty($createdPrinterSn)) {
                $xpyun = new XpyunService();
                $xpyun->delPrintersBulk([$createdPrinterSn]);
            }
        } catch (\Throwable $t) {}
        return ['code' => 1,'message' => $ex->getMessage(), 'data'=>[]];
    }
}
public function read($id) {
    $equipment = $this->model->readById($id);
    if($equipment) {
        $equipment['type_cn'] = empty($this->typeArr[$equipment['type']]) ? '' : $this->typeArr[$equipment['type']];
        $equipment['status_cn'] = empty($this->statusArr[$equipment['status']]) ? '' : $this->statusArr[$equipment['status']];
    }
    return $equipment;
}
```

```
public function update($params, $currentUser) {
    $datetime = date("Y-m-d H:i:s");
    $operator = $currentUser['name'];
    try {
        $equipmentId = $params['id'];
        $equipment = $this->model->readById($equipmentId);
        if(empty($equipment)) {
            throw new Exception("设备不存在");
        }
        if ((int)$equipment['type'] !== 11) {
            if (isset($params['type']) && (int)$params['type'] === 11) {
                throw new Exception('非打印机设备禁止修改为 Xprinter 打印机');
            }
        }
        if ((int)$equipment['type'] === 11) {
            if (isset($params['type']) && (int)$params['type'] !== 11) {
                throw new Exception('Xprinter 打印机设备禁止修改设备类型');
            }
            if (isset($params['sn'])) {
                $newSn = trim((string)$params['sn']);
                $oldSn = isset($equipment['sn']) ? (string)$equipment['sn'] : '';
                if ($newSn !== $oldSn) {
                    throw new Exception('Xprinter 打印机设备禁止修改设备 SN');
                }
            }
        }
        if(isset($params['code'])) {
            $saveParams['code'] = $params['code'];
        }
        if($params['code'] && ($equipment['code'] != $params['code'])) {
            $code_exist = $this->model->inquiryCount(['code'=>$params['code'],'del_flag'=>0]);
            if($code_exist) {
                throw new Exception("设备编号已存在");
            }
        }
        if(isset($params['name'])) {
            $saveParams['name'] = $params['name'];
        }
        if(isset($params['name']) && ($equipment['name'] != $params['name'])) {
            $name_exist = $this->model->inquiryCount(['name'=>$params['name'],'del_flag'=>0]);
            if($name_exist) {
                throw new Exception("设备名称已存在");
            }
        }
        if (isset($params['sn'])) {
            $saveParams['sn'] = $params['sn'];
        }
        if (isset($params['address'])) {
            $saveParams['address'] = trim((string)$params['address']);
        }
```

```
        if(isset($params['type'])) {
            $saveParams['type'] = $params['type'];
        }
        $saveParams['update_time'] = $datetime;
        $saveParams['update_by'] = $operator;
        $saveParams['restaurant_id'] = isset($params['restaurant_id'])?$params['restaurant_id']:'1';
        $saveParams['model'] = isset($params['model'])?$params['model']:1;
        $update = $this->model->modifyById($equipmentId, $saveParams);
        if(!$update) {
            throw new Exception("操作失败");
        }
        $new = $this->model->readById($equipmentId);
        return ['code' => 0,'message' => '操作成功', 'data'=>$new];
    } catch(Exception $ex) {
        return ['code' => 1,'message' => $ex->getMessage(), 'data'=>[]];
    }
}
public function del($ids, $currentUser) {
    try {
        if (empty($ids)) {
            throw new Exception("参数错误");
        }
        $idArr = array_filter(explode(",", $ids));
        if (empty($idArr)) {
            throw new Exception("参数错误");
        }
        $toDelete = $this->model->inquiryAll([
            'id' => ['in', $idArr],
            'del_flag' => 0,
        ], [], ['id', 'name', 'type', 'sn']);
        $printerSns = [];
        $printerItemsForRollback = [];
        if (!empty($toDelete)) {
            foreach ($toDelete as $row) {
                if ((int)$row['type'] === 11) {
                    $sn = isset($row['sn']) ? trim((string)$row['sn']) : '';
                    if ($sn !== '') {
                        $printerSns[] = $sn;
                        $printerItemsForRollback[] = ['sn' => $sn, 'name' => (string)$row['name']];
                    }
                }
            }
        }
        Db::startTrans();
        $params = [];
        $params['del_flag'] = 1;
        $params['update_time'] = date("Y-m-d H:i:s");
        $params['update_by'] = $currentUser['name'];
        $params['code'] = Db::raw("CONCAT(code, '_delete')");
        $del = $this->model->modifyByWhere(['id' => ['in', $idArr]], $params);
```

```
        if ($del === false) {
            throw new Exception("删除失败");
        }
        $remoteDeleted = false;
        if (!empty($printerSns)) {
            $xpyun = new XpyunService();
            $xpRes = $xpyun->delPrintersBulk($printerSns);
            if (!isset($xpRes['code']) || (int)$xpRes['code'] !== 0) {
                $msg = isset($xpRes['message']) ? (string)$xpRes['message'] : '删除打印机失败';
                throw new Exception($msg);
            }
            $data = isset($xpRes['data']) ? $xpRes['data'] : null;
            $successList = (is_array($data) && isset($data['success']) && is_array($data['success'])) ? $data['success'] : [];
            foreach ($printerSns as $sn) {
                if (!in_array($sn, $successList, true)) {
                    $failReason = '';
                    if (is_array($data) && isset($data['failMsg']) && is_array($data['failMsg'])) {
                        foreach ($data['failMsg'] as $fm) {
                            if (is_string($fm) && strpos($fm, $sn . ':') === 0) {
                                $failReason = substr($fm, strlen($sn) + 1);
                                break;
                            }
                        }
                    }
                    $reasonMap = [
                        '1001' => '打印机编号和用户不匹配',
                        '1002' => '打印机未注册',
                        '1009' => '删除打印机时编号不能为空',
                        '1010' => '打印机设备编号无效',
                        '1013' => '删除打印设备失败',
                    ];
                    $msg = '删除打印机失败';
                    if ($failReason !== '') {
                        $msg = isset($reasonMap[$failReason]) ? $reasonMap[$failReason] : ('删除打印机失败（错误码 ' . $failReason . ' ）');
                    }
                    throw new Exception($msg);
                }
            }
            'left_dishes' => $left_dishes,
            'right_dishes' => $right_dishes,
        ];
    }
    return $equipments;
}
public function pushDishes($equipment_id, $left_dishes_uuid, $right_dishes_uuid, $currentUser) {
    $operator = $currentUser['name'];
    $weighType  = EquipmentTypeEnum::weighType();
    try {
        $where = [
            'id' => $equipment_id,
```

```
            'del_flag' => 0,
            'type' => ['in', $weighType]
        ];
        $equipment = $this->model->inquiryOne($where);
        if (empty($equipment)) {
            throw new Exception('设备不存在');
        }
        $updateData = [
            'left_dishes_uuid' => $left_dishes_uuid,
            'right_dishes_uuid' => $right_dishes_uuid,
            'push_dishes_user' => $operator,
            'push_dishes_time' => date('Y-m-d H:i:s'),
        ];
        $pushData = [
            'equipment_id' => $equipment_id,
            'type' => 1,
            'left_dishes_uuid' => $left_dishes_uuid,
            'left_dishes_weight' => $equipment['left_dishes_weight'],
            'right_dishes_uuid' => $right_dishes_uuid,
            'right_dishes_weight' => $equipment['right_dishes_weight'],
            'create_by' => $operator,
        ];
        if ($left_dishes_uuid != $equipment['left_dishes_uuid']) {
            $updateData['left_dishes_weight_max'] = $equipment['left_dishes_weight'];
        }
        if ($right_dishes_uuid != $equipment['right_dishes_uuid']) {
            $updateData['right_dishes_weight_max'] = $equipment['right_dishes_weight'];
        }
        $this->model->modifyById($equipment_id, $updateData);
        $EquipmentPushDishesModel = new EquipmentPushDishesModel;
        $EquipmentPushDishesModel->save($pushData);
        $this->pushDishesMsg($equipment, $left_dishes_uuid, $right_dishes_uuid);
        return ['code' => 0, 'message' => '排菜完成'];
    } catch (Exception $e) {
        return ['code' => 1, 'message' => $e->getMessage()];
    }
}
public function pushDishesMsg($equipment, $left_dishes_uuid, $right_dishes_uuid) {
    if ($equipment['mate_flag'] != 1 || $equipment['hearttime'] < time() - 60) {
        \think\Log::write('称重台不在线');
        return true;
    }
    $dishes_uuids = [$left_dishes_uuid, $right_dishes_uuid];
    $dishes_uuids = array_filter(array_unique($dishes_uuids));
    $DishesModel = new DishesModel();
    $dishess = $DishesModel->inquiryAll(['uuid' => ['in', $dishes_uuids]], [], ['uuid', 'name', 'category_one', 'file_path']);
    $dishess = array_column($dishess, null, 'uuid');
    $DishesCate = DishesCateModel::getAllData(['pid' => 0]);
    $DishesCate = array_column($DishesCate, null, 'id');
    $dishes_msg = [];
```

```
$dishesInfos = '';
$left_dishes = $dishess[$left_dishes_uuid] ?? [];
if (!empty($left_dishes)) {
    $categoryInfo = $DishesCate[$left_dishes['category_one']];
    $dishes_msg[] = [
        'position' => 1,
        'dishinfo' => [
            'uuid' => $left_dishes['uuid'],
            'name' => $left_dishes['name'],
            'file_path' => getImgFullPath($left_dishes['file_path']),
            'category' => $categoryInfo['ori_id'],
            'category_name' => $categoryInfo['name']
        ]
    ];
    $dishesInfos = $left_dishes['name'];
}
$right_dishes = $dishess[$right_dishes_uuid] ?? [];
if (!empty($right_dishes)) {
    $categoryInfo = $DishesCate[$right_dishes['category_one']];
    $dishes_msg[] = [
        'position' => 2,
        'dishinfo' => [
            'uuid' => $right_dishes['uuid'],
            'name' => $right_dishes['name'],
            'file_path' => getImgFullPath($right_dishes['file_path']),
            'category' => $categoryInfo['ori_id'],
            'category_name' => $categoryInfo['name']
        ]
    ];
    if (empty($dishesInfos)) {
        $dishesInfos = $right_dishes['name'];
    } else {
        $dishesInfos .= ','.$right_dishes['name'];
    }
}
$msg_uuid = guid();
$msg_type = 2;
$msg = array(
    'type' => $msg_type,
    'dishes' => $dishes_msg,
    'msg_uuid' => $msg_uuid,
);
$message = json_encode($msg, JSON_UNESCAPED_UNICODE | JSON_UNESCAPED_SLASHES);
$publish_time = date('Y-m-d H:i:s');
$mq_message_log = array(
    'type' => $msg_type,
    'publish_device_id' => $publish_device_id,
    'receive_device_id' => $equipment['code'],
    'message' => $message,
    'publish_time' => $publish_time,
```

```
            'create_time' => date('Y-m-d H:i:s'),
        );
        $mq_message = array(
            'msg_uuid'          => $msg_uuid,
            'type'              => $msg_type,
            'publish_device_id' => $publish_device_id,
            'receive_device_id' => $equipment['code'],
            'message'           => $message,
            'publish_time'      => $publish_time,
            'publish_num'       => 1,
            'staff_uuid'        => '',
            'status'            => 1,
            'create_time'       => date('Y-m-d H:i:s'),
        );
        $mqMessageLogModel = new MqMessageLog();
        $mqMessageModel = new MqMessage();
        $mqMessageLogModel->add($mq_message_log);
        $mqMessageModel->add($mq_message);
        $msg = [
            '客户' => config('business_name'),
            '设备 id' => $equipment['id'],
            '设备编号' => $equipment['code'],
            '设备名称' => $equipment['name'],
            '菜品信息' => $dishesInfos,
        ];
        \app\common\QwRobot::pushMsgFormat($msg, '称重台排菜消息发送成功');
        return true;
    }
    public function syncDishes($dishes_uuid) {
        $weighType  = EquipmentTypeEnum::weighType();
        $where = [
            'left_dishes_uuid' => $dishes_uuid,
            'del_flag' => 0,
            'mate_flag' => 1,
            'hearttime' => ['>=', time() - 60],
            'type' => ['in', $weighType]
        ];
        $equipments = $this->model->inquiryAll($where);
        if (empty($equipments)) {
            return true;
        }
        foreach ($equipments as $equipment) {
            $this->pushDishesMsg($equipment, $equipment['left_dishes_uuid'], $equipment['right_dishes_uuid']);
        }
        return true;
    }
    public function plateList($where=[], $page=1, $pageSize=15, $orderBy=[], $field=['*']) {
        $equipments = $this->model->inquiry($where, $page, $pageSize, $orderBy, $field);
        if (empty($equipments)) {
            return [];
```

```
        }
        return $equipments;
    }
    public function plateDetail($equipment_id, $menu_date) {
        $data = [
            'id' => $equipment_id,
            'menu_date' => $menu_date,
            'code' => '',
            'name' => '',
            'type' => 8,
            'morning_file_id' => 0,
            'morning_file_url' => '',
            'noon_file_id' => 0,
            'noon_file_url' => '',
            'night_file_id' => 0,
            'night_file_url' => '',
            'night_snack_file_id' => 0,
            'night_snack_file_url' => ''
        ];
        $equipment = $this->model->inquiryOne(['id' => $equipment_id, 'del_flag' => 0], [], ['id','code','name', 'type']);
        if (empty($equipment)) {
            return $data;
        }
        $data = array_merge($data, $equipment);
        $PlateDishes = PlateDishes::detail(['equipment_id' => $equipment_id, 'menu_date' => $menu_date, 'is_delete' => 0]);
        if (empty($PlateDishes)) {
            return $data;
        }
        $file_ids = array_unique(array_filter([$PlateDishes['morning_file_id'], $PlateDishes['noon_file_id'], $PlateDishes['night_file_id'], $PlateDishes['night_snack_file_id']]));
        if (empty($file_ids)) {
            return $data;
        }
        $UploadFiles = UploadFile::getAllData(['id' => ['in', $file_ids]], 'id,file_path');
        if (empty($UploadFiles)) {
            return $data;
        }
        $UploadFiles = array_column($UploadFiles, null, 'id');
        $morning_file = $UploadFiles[$PlateDishes['morning_file_id']] ?? null;
        $data['morning_file_id'] = $morning_file['id'] ?? 0;
        $data['morning_file_url'] = $morning_file['file_url'] ?? '';
        $noon_file = $UploadFiles[$PlateDishes['noon_file_id']] ?? null;
        $data['noon_file_id'] = $noon_file['id'] ?? 0;
        $data['noon_file_url'] = $noon_file['file_url'] ?? '';
        $night_file = $UploadFiles[$PlateDishes['night_file_id']] ?? null;
        $data['night_file_id'] = $night_file['id'] ?? 0;
        $data['night_file_url'] = $night_file['file_url'] ?? '';
        $night_snack_file = $UploadFiles[$PlateDishes['night_snack_file_id']] ?? null;
        $data['night_snack_file_id'] = $night_snack_file['id'] ?? 0;
        $data['night_snack_file_url'] = $night_snack_file['file_url'] ?? '';
        return $data;
```

```
    }
    public function plateUpdate($params, $currentUser) {
        $operator = $currentUser['name'];
        $equipment = $this->model->inquiryOne(['id' => $params['equipment_id'], 'del_flag' => 0], [], ['id','code','name', 'type']);
        if (empty($equipment)) {
            return ['code' => 1, 'message' => '设备不存在', 'data' => []];
        }
        $PlateDishes = PlateDishes::detail(['equipment_id' => $params['equipment_id'], 'menu_date' => $params['menu_date'], 'is_delete' => 0]);
        if (empty($PlateDishes)) {
            $PlateDishes = new PlateDishes();
            $params['create_by'] = $operator;
            $PlateDishes->save($params);
        } else {
            $PlateDishes->save($params);
        }
        $this->model->modifyById($params['equipment_id'], ['update_time' => date('Y-m-d H:i:s'), 'update_by' => $operator]);
        return ['code' => 0, 'message' => '保存成功', 'data' => []];
    }
    public function consumeList($where = [], $orderBy = [], $field) {
        $equipments = $this->model->inquiryAll($where, $orderBy, $field);
        if (empty($equipments)) {
            return [];
        }
        return $equipments;
    }
}
<?php
namespace app\store\controller\dishes;
use think\response\Json;
use app\base\AdminController;
use app\store\service\dishes\Dishes as DishesService;
class Dishes extends AdminController
{
    protected $beforeActionList = ['checkLogin'];
    public function detail(int $id): Json
    {
        $server = new DishesService;
        $detail = $server->detail($id);
        return $this->renderSuccess(compact('detail'));
    }
    public function list(): Json
    {
        $server = new DishesService;
        $list = $server->getList($this->request->param());
        return $this->renderSuccess(compact('list'));
    }
    public function all(): Json
    {
        $server = new DishesService;
        $list = $server->getAll($this->request->param());
```

```
        return $this->renderSuccess(compact('list'));
    }
    public function add(): Json
    {
        $server = new DishesService;
        if ($server->add($this->postForm())) {
            $id  = $server->id;
            return $this->renderSuccess(compact('id'), '添加成功');
        }
        return $this->renderError($server->getError() ?: '添加失败');
    }
    public function edit(int $id): Json
    {
        $server = new DishesService;
        if ($server->edit($id, $this->postForm())) {
            return $this->renderSuccess('更新成功');
        }
        return $this->renderError($server->getError() ?: '更新失败');
    }
    public function delete(int $id): Json
    {
        $server = new DishesService;
        if ($server->setDelete($id)) {
            return $this->renderSuccess('删除成功');
        }
        return $this->renderError($server->getError() ?: '删除成功');
    }
    public function cate(int $pid = -1): Json
    {
        $server = new DishesService;
        $list = $server->cate($pid);
        return json(['code' => 0, 'message' => '查询成功', 'data' => $list]);
    }
    public function dishLibrary()
    {
        $server = new DishesService;
        $list = $server->dishLibrary($this->request->param());
        return $this->renderSuccess(compact('list'));
    }
}
    public function getList(array $param = []): array
    {
        $model = new DishesModel();
        $field = 'id,name,category,weight,energy,carbohydrate,protein,fat,file_id,create_by,create_time';
        $list = $model->getList($param, $field);
        if (empty($list['data'])) {
            return $list;
        }
        $categorys = DishesCateModel::getAllData(['pid' => 0]);
        $categorys = array_column($categorys, null, 'id');
```

```
        $file_ids = array_column($list['data'], 'file_id');
        $files = UploadFileModel::getAllData(['file_id' => $file_ids]);
        $files = array_column($files, null, 'file_id');
        $list['data'] = array_map(function ($item) use ($categorys, $files) {
            $item['category_name'] = $categorys[$item['category']]['name'] ?? '';
            $item['file_url'] = $files[$item['file_id']]['preview_url'] ?? '';
            return $item;
        }, $list['data']);
        return $list;
    }
    public function getAll(array $param = []): \think\Collection
    {
        $model = new DishesModel();
        return $model->getAll($param);
    }
    public function add(array $data, string $create_by = '', string $origin = ''): bool
    {
        $this->validate($data);
        if (DishesModel::checkExist($data['name'])) {
            $this->error = '菜品名称已存在';
            return false;
        }
        $create_by = $create_by ?: $this->getOperatorName();
        $model = new DishesModel;
        $res = $model->add($data, $create_by, $origin);
        $this->id = $model['id'] ?: 0;
        $this->uuid = $model['uuid'] ?: '';
        $this->ai_dishes_id = $model['ai_dishes_id'] ?: 0;
        if ($res) {
            $this->pushLockLygFoodInfo('add', [
                'id' => $this->uuid,
                'foodName' => $model['name'],
            ], $create_by);
        }
        return $res;
    }
    public function detail(int $id)
    {
        $with = ['cate', 'file', 'unsuitable', 'foods', 'tag'];
        $detail = DishesModel::detail($id, $with);
        if (empty($detail)) {
            throwError('数据不存在');
        }
        $detail = $detail->toArray();
        $detail['category_name'] = $detail['cate']['name'] ?? '';
        unset($detail['cate']);
        $detail['file_url'] = $detail['file']['preview_url'] ?? '';
        if (!empty($detail['unsuitable'])) {
            $detail['unsuitable'] = array_column($detail['unsuitable'], 'meal_times');
        }
```

```
    $cookInfo = ConfigEnumModel::getEnumOne((string)EnumTypeEnum::ENUM4, (string)$detail['cook_method']);
    $detail['cook_method_name'] = $cookInfo['enum_name'] ?? '';
    if (!empty($detail['foods'])) {
        $foods = [];
        foreach ($detail['foods'] as $food) {
            $item = [
                'foods_id' => $food['pivot']['foods_id'],
                'foods_weight' => $food['pivot']['foods_weight'],
                'foods_ratio' => $food['pivot']['foods_ratio'],
                'foods_status' => $food['foods_status'],
                'foods_status_name' => FoodsStatus::getName($food['foods_status']),
                'foods_name' => $food['name'],
            ];
            $foods[] = $item;
        }
        $detail['foods'] = $foods;
    }
    return $detail;
}
public function edit(int $id, array $data, string $create_by = ''): bool
{
    $this->validate($data);
    $create_by = $create_by ?: $this->getOperatorName();
    $model = DishesModel::detail($id);
    if (empty($model)) {
        throwError('数据不存在');
    }
    if ($model['name'] !== $data['name'] && DishesModel::checkExist($data['name'])) {
        $this->error = '菜品名称已存在';
        return false;
    }
    $res = $model->edit($data, $create_by);
    if (!$res) {
        return false;
    }
    $EquipmentLogic = new EquipmentLogic();
    $EquipmentLogic->syncDishes($model['uuid']);
    $this->pushLockLygFoodInfo('upd', [
        'id' => $model['uuid'],
        'foodName' => $data['name'],
    ], $create_by);
    return true;
}
public function setDelete(int $id): bool
{
    $model = DishesModel::detail($id);
    if (empty($model)) {
        throwError('数据不存在');
    }
    $create_by = $this->getOperatorName() ?: (string)$model['create_by'];
```

```
        $baiduDishes = new BaiduDishes;
        $imgs = DishesImgModel::getAllData(['dishes_id' => $id, 'is_delete' => 0]);
        if (!empty($imgs)) {
            foreach ($imgs as $item) {
                if (!empty($item['cont_sign'])) {
                    $baiduDishes->delete($item['cont_sign']);
                }
            }
        }
        $res = $model->setDelete();
        if (!$res) {
            throwError('删除失败');
        }
        $img_ids = array_column($imgs, 'id');
        DishesImgModel::updateBase(['is_delete' => 1], ['id' => $img_ids]);
        $this->pushLockLygFoodInfo('del', [
            'id' => $model['uuid'],
            'foodName' => $model['name'],
        ], $create_by);
        return true;
<?php
namespace app\p\logic;
use app\model\Dishes as DishesModel;
use app\model\DishesCapeUuid as DishesCapeUuidModel;
use app\model\Recipes as RecipesModel;
use app\model\Excel as ExcelModel;
use think\Exception;
class Dishes{
    private $model;
    private $dishesCapeUuidModel;
    private $recipesModel;
    private $excel;
    public function __construct() {
        $this->model = new DishesModel();
        $this->dishesCapeUuidModel = new DishesCapeUuidModel();
        $this->recipesModel = new RecipesModel();
        $this->excel = new ExcelModel();
    }
    public function listUuid() {
        return $result = $this->model->inquiryColumn(['is_del'=>1],'uuid');
    }
    public function uuidFind($uuid) {
        return $result = $this->model->read($uuid);
    }
    public function dishesCape($uuid,$field=null) {
        if ($field == null) {
            $result = $this->dishesCapeUuidModel->where(['dishes_uuid'=>$uuid])->find();
        } else {
            $result = $this->dishesCapeUuidModel->where(['cape_uuid'=>$uuid])->find();
        }
```

```
        return $result;
    }
    public function addDisherCape($array) {
        return $this->dishesCapeUuidModel->addDishesCape($array);
    }
    public function recipesUuid($where) {
        return $this->recipesModel->inquiryColumn($where,'dishes_uuid');
    }
    public function dishesCapeUuid($where){
        return $this->dishesCapeUuidModel->whereIn('dishes_uuid',$where)->column('cape_uuid');
    }
    public function foodNutritionExport() {
        try {
            $dishes_result = $this->model->idInquiryAll();
            array_unshift($dishes_result,['序号','菜品名称','能量（Kcal）','碳水化合物（g）','脂肪（g）','蛋白质（g）','膳食纤维（g）','维生素 A（μgRE）','维生素 C（mg）','维生素 E（mg）','维生素 B1（mg）','维生素 B2（mg）','烟酸（mg）','胆固醇（mg）','镁（mg）','钙（mg）','铁（mg）','锌（mg）','铜（mg）','锰（mg）','钾（mg）','硒（mg）','钠（mg）']);
            array_unshift($dishes_result,['序号','菜品名称','宏量元素','','','','微量元素','','','','','','','','','','','']);
            array_unshift($dishes_result,['菜品营养（每 100g 菜品营养素提供值）']);
            $this->excel->foodNutritionExport('菜品营养', $dishes_result);
        } catch (Exception $e) {
            return ['code' => 1, 'message' => $e->getMessage(), 'data' => []];
        }
        return ['code' => 0, 'message' => '导出成功', 'data' =>[]];
    }
}
<?php
namespace app\model;
use think\Model;
use think\Db;
use app\model\BaseModel;
class Dishes extends BaseModel{
    public function recipes($where, $field)
    {
        $result = $this
            ->where($where)
            ->field($field)
            ->find();
        return $result;
    }
    public function idInquiryAll() {
        $sql = "SELECT (@i:=@i+1) id,NAME,round(energy/weight*100,2) as energy,round(carbohydrate/weight*100,2) as carbohydrate,round(fat/weight*100,2) as
fat,round(protein/weight*100,2) as protein,round(dietary_fiber/weight*100,2) as dietary_fiber,round(va/weight*100,2) as va,round(vc/weight*100,2) as vc,round(ve/weight*100,2) as
ve,round(vb1/weight*100,2) as vb1,round(vb2/weight*100,2) as vb2,round(vpp/weight*100,2) as vpp,round(cholesterol/weight*100,2) as cholesterol,round(magnesium/weight*100,2) as
magnesium,round(calcium/weight*100,2) as calcium,round(iron/weight*100,2) as iron,round(zinc/weight*100,2) as zinc,round(cuprum/weight*100,2) as
cuprum,round(manganese/weight*100,2) as manganese,round(potassium/weight*100,2) as potassium,round(selenium/weight*100,2) as selenium,round(sodium/weight*100,2) as sodium
FROM ydy_dishes ,(select @i:=0) as it where status = 1 and is_del = 1 ORDER BY create_time DESC";
        return Db::query($sql);
    }
    public function selectDish($dishArr) {
        $info = $this
            ->alias('a')
            ->field('a.weight, a.ssb, b.food_weight, b.dishes_uuid')
            ->join('ydy_dish b', 'b.dishes_uuid = a.uuid', 'RIGHT')
            ->whereIn('a.uuid', $dishArr)
            ->select()
            ->toArray();
```

```
        return $info;
    }
    public function foodCookMethod($where, $field="*") {
        $result = $this
            ->alias('a')
            ->join('ydy_dish b','a.uuid = b.dishes_uuid')
            ->where($where)
            ->field($field)
            ->select()
            ->toArray();
        return $result;
    }
    public function foodCookMethodRadio($where, $field="*") {
        $result = $this
            ->alias('a')
            ->join('ydy_dish b','a.uuid = b.dishes_uuid')
            ->where($where)
            ->field($field)
            ->group('a.uuid')
            ->select()
            ->toArray();
        return $result;
    }
    public function dishesSelectForScreen($where, $field="*",$order='') {
        $result = $this
            ->alias('a')
            ->join('ydy_dish b','a.uuid = b.dishes_uuid')
            ->where($where)
            ->field($field)
            ->group('a.uuid')
            ->order($order)
            ->select()
            ->toArray();
        return $result;
    }
}
<?php
namespace app\model;
class DishesRecognizeImg extends BaseModel
{
}
<?php
namespace app\model;
class DishesHardwareImg extends BaseModel
{
}
<?php
namespace app\model;
use app\model\BaseModel;
use think\Exception;
```

```
class Equipment extends BaseModel {
    public function __construct($data = array()) {
        parent::__construct($data);
    }
}
<?php
namespace app\api\service;
use app\api\model\Equipment as EquipmentModel;
use app\common\model\UploadFile;
use app\common\model\PlateDishes;
use app\common\service\BaseService;
use app\cores\exception\BaseException;
use think\App;
class Plate extends BaseService
{
    private function plateInfo($code)
    {
        $where = [
            'code' => $code,
            'del_flag' => 0,
            'type' => 8
        ];
        $weighInfo = EquipmentModel::get($where);
        if (empty($weighInfo)) {
            throwError('设备不存在');
        }
        return $weighInfo;
    }
    public function dishes($code)
    {
        $data = [
            'img_url' => ''
        ];
        $plateInfo = $this->plateInfo($code);
        $menu_date = date('Y-m-d');
        $menu_time = date('Y-m-d H:i:s');
        $PlateDishes = PlateDishes::detail(['equipment_id' => $plateInfo['id'], 'menu_date' => $menu_date, 'is_delete' => 0]);
        if (empty($PlateDishes)) {
            return $data;
        }
        $meal_times = \app\common\Common::mealTimes($plateInfo['restaurant_id'], $menu_time)['meal_times'];
        $file_id = 0;
        if ($meal_times == 1) {
            $file_id = $PlateDishes['morning_file_id'];
        } elseif ($meal_times == 2) {
            $file_id = $PlateDishes['noon_file_id'];
        } elseif ($meal_times == 3) {
            $file_id = $PlateDishes['night_file_id'];
        } elseif ($meal_times == 4) {
            $file_id = $PlateDishes['night_snack_file_id'];
```