<**style lang="less" scoped**>  
 .**background** {  
 **background-color**: **#808080**;  
 **margin**: 0;  
 **padding**: 10**px**;  
 }  
 .**info-box** {  
 **height**: **calc**(~**"100vh - 125px"**);  
 **overflow-y**: **auto**;  
 }  
</**style**>  
  
<**template**>  
 <**div**>  
 <**default-page-box**>  
 <**template slot="tools-bar"**>  
 <**Button type="primary" size="small" @click="**submitForm**"**><**i class="iconfont icon-save"**></**i**> 保存</**Button**>  
 </**template**>  
 <**template slot="content"**>  
 <**Row**>  
 <**i-col span="12" class="info-box" v-loading="loading"**>  
 <**Card class="card-content"**>  
 <**Tabs type="line" size="small" :animated="**false**"**>  
 <**TabPane label="下货纸" name="xhzTab"** >  
 *<!--收发通-->* <**Card class="customer-card" dis-hover** >  
 <**Row class="vertical-split-row border-bottom-line"**>  
 <**i-col span="12"**>  
 <**card-header**>  
 <**Row**>  
 <**i-col span="8"**>  
 <**div**>{{this.$t('seaShip.shipperTitle')}}</**div**>  
 </**i-col**>  
 <**i-col span="15"**>  
 <**Input v-model="seaShip**.**shipperCode" @on-focus="**onFieldFocus('shipperCode')**" @on-blur="**onFieldBlur**" size="small" ref="shipperCode" type="text" :placeholder="**$t('seaShip.codeTag')**" @on-enter="**$refs.**notifyCode**.focus()**" :maxlength=**50 ></**Input**>  
 </**i-col**>  
 </**Row**>  
 <**div slot="func"**>  
 <**Button class="header-btn" type="default" size="small" @click="**clickSelectParty('shipper','shipper')**"**>+</**Button**>  
 </**div**>  
 </**card-header**>  
 <**Form :label-width="**40**" ref="formShipperValidate" :model="seaShip"**>  
 <**FormItem :label="**$t('seaShip.nameTag')**" prop="shipper"** >  
 <**Input v-model="seaShip**.**shipper" @on-focus="**onFieldFocus('shipper')**" @on-blur="**onFieldBlur**" class="text-uppercase" size="small" type="textarea" ref="shipper" :placeholder="**$t('seaShip.nameTag')**" :rows="**2**" :maxlength=**255></**Input**>  
 </**FormItem**>  
 <**FormItem :label="**$t('seaShip.addressTag')**" prop="shipAddr"** >  
 <**Input v-model="seaShip**.**shipAddr" @on-focus="**onFieldFocus('shipAddr')**" @on-blur="**onFieldBlur**" class="text-uppercase" size="small" type="textarea" :placeholder="**$t('seaShip.addressTag')**" :rows="**4**" :maxlength=**255></**Input**>  
 </**FormItem**>  
 <**FormItem :label="**$t('seaShip.TelephoneTag')**" prop="shipTel"**>  
 <**Input v-model="seaShip**.**shipTel" @on-focus="**onFieldFocus('shipTel')**" @on-blur="**onFieldBlur**" size="small" :placeholder="**$t('seaShip.TelephoneTag')**" :maxlength=**255></**Input**>  
 </**FormItem**>  
 </**Form**>  
 </**i-col**>  
 <**i-col span="12"**>  
 <**card-header**>  
 <**Row**>  
 <**i-col span="8"**>  
 <**div**>{{this.$t('seaShip.notifyPartyTitle')}}</**div**>  
 </**i-col**>  
 <**i-col span="15"**>  
 <**Input v-model="seaShip**.**notifyCode" @on-focus="**onFieldFocus('notifyCode')**" @on-blur="**onFieldBlur**" size="small" ref="notifyCode" type="text" :placeholder="**$t('seaShip.codeTag')**" @on-enter="**$refs.**consigneeCode**.focus()**" :maxlength=**50></**Input**>  
 </**i-col**>  
 </**Row**>  
 <**div slot="func"**>  
 <**Button class="header-btn" type="default" size="small" @click="**clickSelectParty('consigneeOrNotifyParty','notifyPartyFlag')**"**>+</**Button**>  
 </**div**>  
 </**card-header**>  
 <**Form :label-width="**40**" ref="formNotifyValidate" :model="seaShip"**>  
 <**FormItem :label="**$t('seaShip.nameTag')**" prop="notifyParty"**>  
 <**Input v-model="seaShip**.**notifyParty" @on-focus="**onFieldFocus('notifyParty')**" @on-blur="**onFieldBlur**" class="text-uppercase" size="small" type="textarea" :placeholder="**$t('seaShip.nameTag')**" :rows="**2**" :maxlength=**255></**Input**>  
 </**FormItem**>  
 <**FormItem :label="**$t('seaShip.addressTag')**" prop="notifyAddr"**>  
 <**Input v-model="seaShip**.**notifyPartyAddr" @on-focus="**onFieldFocus('notifyPartyAddr')**" @on-blur="**onFieldBlur**" class="text-uppercase" size="small" type="textarea" :placeholder="**$t('seaShip.addressTag')**" :rows="**4**" :maxlength=**255></**Input**>  
 </**FormItem**>  
 <**FormItem :label="**$t('seaShip.TelephoneTag')**"**>  
 <**Input v-model="seaShip**.**notifyPartyTel" @on-focus="**onFieldFocus('notifyPartyTel')**" @on-blur="**onFieldBlur**" size="small" ref="notifyTel" :placeholder="**$t('seaShip.TelephoneTag')**" :maxlength=**255></**Input**>  
 </**FormItem**>  
 </**Form**>  
 </**i-col**>  
 </**Row**>  
 <**Row class="vertical-split-row "**>  
 <**i-col span="12"**>  
 <**card-header**>  
 <**Row**>  
 <**i-col span="8"**>  
 <**div**>{{this.$t('seaShip.consigneeTitle')}}</**div**>  
 </**i-col**>  
 <**i-col span="15"**>  
 <**Input v-model="seaShip**.**consigneeCode" @on-focus="**onFieldFocus('consigneeCode')**" @on-blur="**onFieldBlur**" size="small"  
 type="text"  
 ref="consigneeCode"  
 @on-enter="**$refs.**notifyParty2Code**.focus()**"  
 :placeholder="**$t('seaShip.codeTag')**"  
 :maxlength=**50></**Input**>  
 </**i-col**>  
 </**Row**>  
 <**div slot="func"**>  
 <**Button class="header-btn" type="default" size="small"  
 @click="**clickSelectParty('consigneeOrNotifyParty','consignee')**"**>  
 +  
 </**Button**>  
 </**div**>  
 </**card-header**>  
 <**Form :label-width="**40**" ref="formConsigneValidate" :model="seaShip"**>  
 <**FormItem :label="**$t('seaShip.nameTag')**" prop="consignee"**>  
 <**Input v-model="seaShip**.**consignee" @on-focus="**onFieldFocus('consignee')**" @on-blur="**onFieldBlur**" class="text-uppercase" size="small" type="textarea"  
 :placeholder="**$t('seaShip.nameTag')**" :rows="**2**" :maxlength=**255></**Input**>  
 </**FormItem**>  
 <**FormItem :label="**$t('seaShip.addressTag')**" prop="consAddr"**>  
 <**Input v-model="seaShip**.**consAddr" @on-focus="**onFieldFocus('consAddr')**" @on-blur="**onFieldBlur**" class="text-uppercase" size="small" type="textarea"  
 :placeholder="**$t('seaShip.addressTag')**"  
 :rows="**4**" :maxlength=**255></**Input**>  
 </**FormItem**>  
 <**FormItem :label="**$t('seaShip.TelephoneTag')**"** >  
 <**Input v-model="seaShip**.**consTel" @on-focus="**onFieldFocus('consTel')**" @on-blur="**onFieldBlur**" size="small"  
 :placeholder="**$t('seaShip.TelephoneTag')**" :maxlength=**255></**Input**>  
 </**FormItem**>  
 </**Form**>  
 </**i-col**>  
 <**i-col span="12"**>  
 <**card-header**>  
 <**Row**>  
 <**i-col span="8"**>  
 <**div**>{{this.$t('seaShip.notifyParty2')}}</**div**>  
 </**i-col**>  
 <**i-col span="15"**>  
 <**Input v-model="seaShip**.**notifyParty2Code" @on-focus="**onFieldFocus('notifyParty2Code')**" @on-blur="**onFieldBlur**" size="small" type="text" @on-enter="**$refs.**overseaAgentdtsCode**.focus()**" :placeholder="**$t('seaShip.codeTag')**" ref="notifyParty2Code" :maxlength=**50></**Input**>  
 </**i-col**>  
 </**Row**>  
 <**div slot="func"**>  
 <**Button class="header-btn" type="default" size="small" @click="**clickSelectParty('consigneeOrNotifyParty','notifyParty2')**"**>+</**Button**>  
 </**div**>  
 </**card-header**>  
 <**Form :label-width="**40**" ref="formNotifyParty2Validate" :model="seaShip"**>  
 <**FormItem :label="**$t('seaShip.nameTag')**" prop="notifyParty2"**>  
 <**Input v-model="seaShip**.**notifyParty2" @on-focus="**onFieldFocus('notifyParty2')**" @on-blur="**onFieldBlur**" class="text-uppercase" size="small" type="textarea" :placeholder="**$t('seaShip.nameTag')**" :rows="**2**" :maxlength=**255></**Input**>  
 </**FormItem**>  
 <**FormItem :label="**$t('seaShip.addressTag')**" prop="notifyParty2"**>  
 <**Input v-model="seaShip**.**notifyParty2Addr" @on-focus="**onFieldFocus('notifyParty2Addr')**" @on-blur="**onFieldBlur**" class="text-uppercase" size="small" type="textarea" :placeholder="**$t('seaShip.addressTag')**" :rows="**4**" :maxlength=**255></**Input**>  
 </**FormItem**>  
 <**FormItem :label="**$t('seaShip.TelephoneTag')**"**>  
 <**Input v-model="seaShip**.**notifyParty2Tel" @on-focus="**onFieldFocus('notifyParty2Tel')**" @on-blur="**onFieldBlur**" size="small" :placeholder="**$t('seaShip.TelephoneTag')**" :maxlength=**255></**Input**>  
 </**FormItem**>  
 </**Form**>  
 </**i-col**>  
 </**Row**>  
  
 </**Card**>  
 *<!--揽货-->* <**Card class="customer-card" dis-hover**>  
 <**Row**>  
 <**Form :label-width="**100**"**>  
  
 <**i-col :span="**8**"**>  
 <**FormItem label="交货地点"**>  
 <**Input v-model="seaShip**.placeOfReceipt**" size="small" @on-focus="**onFieldFocus('placeOfReceipt')**" @on-blur="**onFieldBlur**"  
 placeholder="交货地点" :maxlength=**64></**Input**>  
 </**FormItem**>  
 </**i-col**>  
 <**i-col :span="**8**"**>  
 <**FormItem label="装货港"**>  
 <**Input v-model="seaShip**.**portOfLoading" size="small" @on-focus="**onFieldFocus('portOfLoading')**" @on-blur="**onFieldBlur**"  
 placeholder="装货港" :maxlength=**64></**Input**>  
 </**FormItem**>  
 </**i-col**>  
  
 <**i-col :span="**8**"**>  
 <**FormItem label="卸货港"**>  
 <**Input v-model="seaShip**.**portOfDischarge" size="small" @on-focus="**onFieldFocus('portOfDischarge')**" @on-blur="**onFieldBlur**"  
 placeholder="卸货港" :maxlength=**64></**Input**>  
 </**FormItem**>  
 </**i-col**>  
  
 <**i-col :span="**8**"**>  
 <**FormItem label="目的港"**>  
 <**Input v-model="seaShip**.**placeOfDelivery" size="small" @on-focus="**onFieldFocus('placeOfDelivery')**" @on-blur="**onFieldBlur**"  
 placeholder="目的港" :maxlength=**64></**Input**>  
 </**FormItem**>  
 </**i-col**>  
  
 <**i-col :span="**8**"**>  
 <**FormItem label="运输条款"**>  
 <**Input v-model="seaShip**.**serviceTerm" size="small" @on-focus="**onFieldFocus('serviceTerm')**" @on-blur="**onFieldBlur**"  
 placeholder="运输条款" :maxlength=**64></**Input**>  
 </**FormItem**>  
 </**i-col**>  
 <**i-col :span="**8**"**>  
 <**FormItem label="付款方式"**>  
 <**Input v-model="seaShip**.**freightCharges" size="small" @on-focus="**onFieldFocus('freightCharges')**" @on-blur="**onFieldBlur**"  
 placeholder="付费方式" :maxlength=**64></**Input**>  
 </**FormItem**>  
 </**i-col**>  
 <**i-col :span="**8**"**>  
 <**FormItem label="约号"**>  
 <**Input v-model="seaShip**.**contractNo" size="small" @on-focus="**onFieldFocus('contractNo')**" @on-blur="**onFieldBlur**"  
 placeholder="约号" :maxlength=**64></**Input**>  
 </**FormItem**>  
 </**i-col**>  
 <**i-col :span="**8**"**>  
 <**FormItem label="HS Code"**>  
 <**Input v-model="seaShip**.**hsCode" size="small" @on-focus="**onFieldFocus('hsCode')**" @on-blur="**onFieldBlur**"  
 placeholder="HS CODE" :maxlength=**64></**Input**>  
 </**FormItem**>  
 </**i-col**>  
 </**Form**>  
 </**Row**>  
 </**Card**>  
 *<!--件重尺-->* <**Card class="customer-card" dis-hover**>  
 <**Form ref="formMarkNoValidate" :model="seaShip"**>  
 <**Row class="vertical-split-row border-bottom-line" style="height: 62px;"**>  
 <**i-col span="6"**>  
 <**card-header**>{{this.$t('seaShip.containerCountTitle')}}</**card-header**>  
 <**Input style="width: 100%" @on-focus="**onFieldFocus('packageSum')**" @on-blur="**onFieldBlur**" v-model="seaShip**.**packageSum" size="small" ref="packageSum" :placeholder="**$t('seaShip.containerCountTitle')**"**></**Input**>  
 </**i-col**>  
 <**i-col span="6"**>  
 <**card-header**>包装种类</**card-header**>  
 <**Input style="width: 100%" @on-focus="**onFieldFocus('packageType')**" @on-blur="**onFieldBlur**" v-model="seaShip**.**packageType" size="small" ref="packageType" placeholder="包装种类"**></**Input**>  
 </**i-col**>  
 <**i-col span="6"**>  
 <**card-header**>{{this.$t('seaShip.grossWeight')}}</**card-header**>  
 <**Input style="width: 100%" @on-focus="**onFieldFocus('grossWeight')**" @on-blur="**onFieldBlur**" v-model="seaShip**.**grossWeight" size="small" :placeholder="**$t('seaShip.grossWeight')**" ref="grossWeight" @on-enter="**$refs.**measurement**.focus()**"** />  
 </**i-col**>  
 <**i-col span="6"**>  
 <**card-header**>{{this.$t('seaShip.measurement')}}</**card-header**>  
 <**Input style="width: 100%" @on-focus="**onFieldFocus('measurement')**" @on-blur="**onFieldBlur**" v-model="seaShip**.**measurement" number size="small" ref="measurement" :placeholder="**$t('seaShip.measurement')**" @on-enter="**$refs.**edi**.focus()**"** />  
 </**i-col**>  
 </**Row**>  
 <**Row class="vertical-split-row border-bottom-line" style="height: 120px;"**>  
 <**i-col span="6"**>  
 <**card-header**>{{this.$t('seaShip.containerQty')}}</**card-header**>  
 <**div**>  
 <**table border="1" style="border: 1px #eeeeee;text-align: center;"**>  
 <**tr**>  
 <**th width="100"**>箱型</**th**>  
 <**th width="50"**>箱量</**th**>  
 </**tr**>  
 <**tr v-for="**(***item***, ***key***) in **seaShip**.**containerInfo" :key="*key*"**>  
 <**td**>{{item.containerCode}}</**td**>  
 <**td**>{{item.containerCount}}</**td**>  
 </**tr**>  
 </**table**>  
 </**div**>  
  
 </**i-col**>  
 </**Row**>  
  
 <**Row class="vertical-split-row"**>  
 <**i-col span="12"**>  
 <**card-header**>{{this.$t('seaShip.markNoTitle')}}</**card-header**>  
 <**FormItem prop="markNo"**>  
 <**Input v-model="seaShip**.**marksNos" @on-focus="**onFieldFocus('marksNos')**" @on-blur="**onFieldBlur**" size="small" ref="markNo" class="marks-nos text-uppercase" :rows="**10**" type="textarea" :placeholder="**$t('seaShip.markNo')**"** />  
 </**FormItem**>  
 </**i-col**>  
 <**i-col span="12" class="pb-10"**>  
 <**card-header**>  
 {{this.$t('seaShip.goodsDescTitle')}}  
 </**card-header**>  
 <**div class="flex"**>  
 <**div class="flex-auto"**>  
 <**FormItem prop="goodsDesc"**>  
 <**Input v-model="seaShip**.**goodsDescription" @on-focus="**onFieldFocus('goodsDescription')**" @on-blur="**onFieldBlur**" size="small" class="description-of-goods text-uppercase" type="textarea" :rows="**10**" :placeholder="**$t('seaShip.goodsDesc')**"** />  
 </**FormItem**>  
 </**div**>  
 </**div**>  
 </**i-col**>  
 </**Row**>  
 </**Form**>  
 </**Card**>  
 </**TabPane**>  
 <**TabPane label="用箱信息" name="boxTab"** >  
 <**sea-ship-container :containerData.sync="containerData"  
 :seaShip.sync="seaShip"  
 :needDeleteLogicContainerIds.sync="needDeleteLogicContainerIds"  
 :goodsData.sync="goodsData" :historyData.sync="historyData" :reloadFlag.sync="reloadFlag"  
 :needDeleteLogicGoodsIds.sync="needDeleteLogicGoodsIds" ref="seaShipContainerRef"**></**sea-ship-container**>  
 </**TabPane**>  
 <**TabPane label="VGM" name="vgm"** >  
 <**sea-ship-vgm :order-no="seaShip**.**orderNo" :vgm-list="vgmList"  
 :bl-no="seaShip**.**blNo" :shipper="seaShip**.**shipper"**></**sea-ship-vgm**>  
 </**TabPane**>  
 </**Tabs**>  
 </**Card**>  
 </**i-col**>  
 <**i-col span="12"**>  
 <**Tag color="green"**>识别率: {{ this.ocrRate }} %</**Tag**>  
 <**ocr-template-box class="flex-auto" @on-change="**onChange**" height="calc(100vh - 125px)" :blockPosition="blockPosition" file-url="abc" :file-body="rawHtmlBody"**></**ocr-template-box**>  
 </**i-col**>  
 </**Row**>  
  
 </**template**>  
 </**default-page-box**>  
 </**div**>  
</**template**>  
  
<**script**>  
**import** DefaultPageBox **from '@/components/page/default-page-box/index'**;  
**import** CardHeader **from '@/components/card/card-header/index'**;  
**import** OcrTemplateBox **from '@/components/template/ocr-template-box/index'**;  
**import** api **from '@/api/ocr/ocrOperation'**;  
**import** SeaShipVgm **from './sea-ship-vgm'**;  
**import** SeaShipContainer **from './seaShipContainer'**;  
**import** ICol **from '../../../../iview/src/components/grid/col'**;  
**import** apiSheet **from '@/api/ocr/ocrSheet'**;  
**import** \_ **from 'lodash'**;  
  
**export default** {  
 **name**: **'export-sea-ship-list'**,  
 **components**: {  
 ICol,  
 DefaultPageBox,  
 CardHeader,  
 OcrTemplateBox,  
 SeaShipVgm,  
 SeaShipContainer  
 },  
 data() {  
 **return** {  
 **ocrRate**: 0,  
 **orderNo**: **''**,  
 **containerData**: [],  
 **needDeleteLogicContainerIds**: [],  
 **goodsData**: [],  
 **historyData**: {},  
 **needDeleteLogicGoodsIds**: [],  
 **vgmList**: [],  
 **rawHtmlBody**: **''**,  
 **loading**: **false**,  
 **schemaList**: [],  
 **currentSchemaList**: [],  
 **blockPosition**: **null**,  
 **file**: **null**,  
 **reloadFlag**: 0,  
 **tableOpt**: {  
 **pageNum**: 1,  
 **pageSize**: 10,  
 **sortList**: [],  
 **airPortFlag**: **'SEASHIPPORT'**,  
 **codeType**: **'PORTCD'**,  
 **searchKey**: **''**,  
 **filterList**: []  
 },  
 **prepaidLocTableData**: {  
 **loading**: **false**,  
 **list**: [],  
 **total**: 0  
 },  
 **preVoyNoTableData**: {  
 **loading**: **false**,  
 **list**: [],  
 **total**: 0  
 },  
 **paybLocTableData**: {  
 **loading**: **false**,  
 **list**: [],  
 **total**: 0  
 },  
 **rcvPortTableData**: {  
 **loading**: **false**,  
 **list**: [],  
 **total**: 0  
 },  
 **loadPortTableData**: {  
 **loading**: **false**,  
 **list**: [],  
 **total**: 0  
 },  
 **transPortTableData**: {  
 **loading**: **false**,  
 **list**: [],  
 **total**: 0  
 },  
 **dischgPortTableData**: {  
 **loading**: **false**,  
 **list**: [],  
 **total**: 0  
 },  
 **podTableData**: {  
 **loading**: **false**,  
 **list**: [],  
 **total**: 0  
 },  
 **finaDestCodeTableData**: {  
 **loading**: **false**,  
 **list**: [],  
 **total**: 0  
 },  
 **routeCodeTableData**: {  
 **loading**: **false**,  
 **list**: [],  
 **total**: 0  
 },  
 **firstCallPortTableData**: {  
 **loading**: **false**,  
 **list**: [],  
 **total**: 0  
 },  
 */\*\* 选择船代的body \*/* **tableOpts**: {  
 **pageNum**: 1,  
 **pageSize**: 10,  
 **sortList**: [],  
 **searchKey**: **''**,  
 **filterList**: []  
 },  
 **seaShip**: {  
 **orderNo**: **''**,  
 **blNo**: **''**,  
 **shipper**: **''**,  
 **shipAddr**: **''**,  
 **shipTel**: **''**,  
 **consignee**: **''**,  
 **consAddr**: **''**,  
 **consTel**: **''**,  
 **notifyParty**: **''**,  
 **notifyPartyAddr**: **''**,  
 **notifyPartyTel**: **''**,  
 **notifyParty2**: **''**,  
 **notifyParty2Addr**: **''**,  
 **notifyParty2Tel**: **''**,  
 **portOfLoading**: **''**,  
 **portOfDischarge**: **''**,  
 **placeOfDelivery**: **''**,  
 **marksNos**: **''**,  
 **goodsDescription**: **''**,  
 **grossWeight**: **''**,  
 **measurement**: **''**,  
 **serviceTerm**: **''**,  
 **freightCharges**: **''**,  
 **containersInfo**: **''**,  
 **contractNo**: **''**,  
 **hsCode**: **''**,  
 **packageSum**: **''**,  
 **packageType**: **''**,  
 **containerInfo**: []  
 },  
 **fieldData**: {}  
 };  
 },  
 **methods**: {  
 submitForm() {  
 **let** body = {  
 **modifyJson**: ***JSON***.stringify(**this**.**seaShip**),  
 **resultJson**: ***JSON***.stringify(**this**.**historyData**),  
 **recognitionRate**: (**this**.**ocrRate** / 100)  
 };  
 */\*\* 识别结果保存到数据库 \*/* apiSheet.createOcrSheet(body).then(({data}) => {  
 **if** (data.returnCode === 200) {  
 ***console***.log(data.returnData);  
 }  
 });  
 */\*\* 识别出的下货纸业务数据保存到数据库 \*/  
 //* ***TODO: 识别出的下货纸业务数据保存到数据库*** },  
  
 strSimilarity2Number (s, t) {  
 **var** n = s.**length**, m = t.**length**, d = [];  
 **var** i, j, s\_i, t\_j, cost;  
 **if** (n == 0) **return** m;  
 **if** (m == 0) **return** n;  
 **for** (i = 0; i <= n; i++) {  
 d[i] = [];  
 d[i][0] = i;  
 }  
 **for** (j = 0; j <= m; j++) {  
 d[0][j] = j;  
 }  
 **for** (i = 1; i <= n; i++) {  
 s\_i = s.charAt(i - 1);  
 **for** (j = 1; j <= m; j++) {  
 t\_j = t.charAt(j - 1);  
 **if** (s\_i == t\_j) {  
 cost = 0;  
 } **else** {  
 cost = 1;  
 }  
 d[i][j] = **this**.Minimum(d[i - 1][j] + 1, d[i][j - 1] + 1, d[i - 1][j - 1] + cost);  
 }  
 }  
 **return** d[n][m];  
 },  
 *// 两个字符串的相似程度，并返回相似度百分比* strSimilarity2Percent (s, t) {  
 **var** l = s.**length** > t.**length** ? s.**length** : t.**length**;  
 **var** d = **this**.strSimilarity2Number(s, t);  
 **return** (1 - d / l).toFixed(4);  
 },  
 Minimum (a, b, c) {  
 **return** a < b ? (a < c ? a : c) : (b < c ? b : c);  
 },  
  
 compare(x, y) {  
 ***console***.log(**'x='** + x);  
 ***console***.log(**'y='** + y);  
 **let** z = 0;  
 **let** s = x.**length** + y.**length**;  
  
 x.sort();  
 y.sort();  
 **let** a = x.shift();  
 **let** b = y.shift();  
  
 **while** (a !== **undefined** && b !== **undefined**) {  
 **if** (a === b) {  
 z++;  
 a = x.shift();  
 b = y.shift();  
 } **else if** (a < b) {  
 a = x.shift();  
 } **else if** (a > b) {  
 b = y.shift();  
 }  
 }  
 **return** z / s \* 200;  
 },  
  
 onFieldFocus(fieldCode) {  
 **if** (**this**.**fieldData**[fieldCode] != **undefined**) {  
 **let** field = **this**.**fieldData**[fieldCode];  
 **this**.**blockPosition** = {  
 **left**: field.**left** / (2480 / 100) + **'%'**,  
 **top**: field.**top** / (3507 / 100) + **'%'**,  
 **width**: (field.**right** - field.**left**) / (2480 / 100) + **'%'**,  
 **height**: (field.**bottom** - field.**top**) / (3507 / 100) + **'%'** };  
 ***console***.log(**'%'** + field.**top** / (3507 / 100) + **','** + field.**left** / (2480 / 100) + **','** + field.**bottom** / (3507 / 100) + **','** + field.**right** / (2480 / 100));  
 }  
 },  
 onFieldBlur() {  
 **this**.**blockPosition** = **null**;  
 },  
 handleCurrentChange(val) {  
 **this**.**modal**.**currentRow** = val;  
 },  
 onChange(file) {  
 **this**.**file** = file;  
 **this**.ocrOperation(file);  
 },  
 ocrOperation() {  
 **this**.**loading** = **true**;  
 **this**.**seaShip** = {  
 **shipper**: **''**,  
 **shipAddr**: **''**,  
 **shipTel**: **''**,  
 **consignee**: **''**,  
 **consAddr**: **''**,  
 **consTel**: **''**,  
 **notifyParty**: **''**,  
 **notifyPartyAddr**: **''**,  
 **notifyPartyTel**: **''**,  
 **notifyParty2**: **''**,  
 **notifyParty2Addr**: **''**,  
 **notifyParty2Tel**: **''**,  
 **portOfLoading**: **''**,  
 **portOfDischarge**: **''**,  
 **placeOfDelivery**: **''**,  
 **marksNos**: **''**,  
 **goodsDescription**: **''**,  
 **grossWeight**: **''**,  
 **measurement**: **''**,  
 **serviceTerm**: **''**,  
 **freightCharges**: **''**,  
 **containersInfo**: **''**,  
 **contractNo**: **''**,  
 **hsCode**: **''**,  
 **packageSum**: **''**,  
 **packageType**: **''**,  
 **containerInfo**: []  
 };  
 **let** file = **this**.**file**;  
 **let** body = {  
 **file**: file  
 };  
 api.ocrOperation(body).then(({data}) => {  
 **if** (data.returnCode === 200) {  
 **if** (data.returnData.**fieldData** != **null**) {  
 **for** (**let** index **in** data.returnData.**fieldData**) {  
 **let** item = data.returnData.**fieldData**[index];  
 **if** (index == **'packageInfo'** && item.dataList != **null** && item.**dataList**.**length** > 0) {  
 **this**.**seaShip**[**'packageType'**] = item.**dataList**[0][**'packageType'**];  
 **this**.**seaShip**[**'packageSum'**] = item.**dataList**[0][**'packageSum'**];  
 } **else if** (index == **'containerInfo'**) {  
 **this**.**seaShip**[**'containerInfo'**] = item.dataList;  
 } **else** {  
 **this**.**seaShip**[index] = item.valueStr;  
 }  
 }  
 **this**.**historyData** = \_.cloneDeep(**this**.**seaShip**);  
 **this**.**fieldData** = data.returnData.**fieldData**;  
 **this**.**rawHtmlBody** = data.returnData.rawHtml;  
 }  
  
 **if** (data.returnData.**data** != **null**) {  
 ***console***.log(data.returnData.**data**);  
 **for** (**let** index **in** data.returnData.**data**.containerList) {  
 **let** curConRow = data.returnData.**data**.containerList[index];  
 **let** curConIdx = (-1 \* index) - 1;  
 **this**.**containerData**.push({  
 **id**: curConIdx,  
 **orderNo**: **this**.**orderNo** || **''**,  
 **containerNo**: curConRow.**containerNo**,  
 **sealNo**: curConRow.**sealNo**,  
 **containerSize**: curConRow.**containerSize**,  
 **containerModel**: curConRow.**containerModel**,  
 **containerNum**: curConRow.**containerNum** });  
  
 **let** rowDetails = curConRow.containerDetails;  
 **for** (**let** jdx **in** rowDetails) {  
 **let** curGoodsRow = rowDetails[jdx];  
 **let** curGoodsIdx = (-1 \* jdx) - 1;  
 **this**.**goodsData**.push({  
 **id**: curGoodsIdx,  
 **containerId**: curConIdx,  
 **orderNo**: **this**.**orderNo** || **''**,  
 **goodsName**: curGoodsRow.**goodsName**,  
 **hscode**: curGoodsRow.**hscode**,  
 **ncm**: curGoodsRow.**ncm**,  
 **mark**: curGoodsRow.**mark**,  
 **pieceNum**: curGoodsRow.**pieceNum**,  
 **packageBox**: curGoodsRow.**packageBox**,  
 **weight**: curGoodsRow.**weight**,  
 **volume**: curGoodsRow.**volume**,  
 **remark**: curGoodsRow.**remark**,  
 **originCertificateNo**: curGoodsRow.**originCertificateNo**,  
 **poNo**: curGoodsRow.**poNo** });  
 }  
  
 **this**.**vgmList**.push({  
 **orderNo**: **this**.**orderNo** || **''**,  
 **blNo**: **this**.**seaShip**[**'blNo'**],  
 **vgmBoxNo**: curConRow.**containerNo**,  
 **vgmScalaMethod**: curConRow.**method**,  
 **vgmWeight**: curConRow.**vgm**,  
 **vgmScalaTime**: **new *Date***()  
 });  
 }  
 }  
 } **else** {  
 **this**.$Message.warning(**'数据异常,请重试！'**);  
 }  
 **this**.**loading** = **false**;  
 });  
 },  
 canvassChange (val) {  
 **const** curCustomerLinker = **this**.**freightTypeList**.filter(**function** (ele, index, array) {  
 **if** (ele.**code** === val) {  
 **return true**;  
 }  
 **return false**;  
 });  
 **if** (curCustomerLinker.**length** > 0) {  
 **this**.**seaShipBase**.**canvassFlagName** = curCustomerLinker[0].**chineseName**;  
 } **else** {  
 **this**.**seaShipBase**.**canvassFlagName** = **''**;  
 }  
 },  
  
 amsChange (val) {  
 **if** (val === **'N'**) {  
 **this**.**seaShip**.**amsStatus** = **'0'**;  
 } **else if** (val === **'Y'**) {  
 **this**.**seaShip**.**amsStatus** = **'1'**;  
 }  
 },  
 aciChange (val) {  
 **if** (val === **'N'**) {  
 **this**.**seaShip**.**aciStatus** = **'0'**;  
 } **else if** (val === **'Y'**) {  
 **this**.**seaShip**.**aciStatus** = **'1'**;  
 }  
 }  
 },  
 **watch**: {  
 **seaShip**: {  
 handler(newValue) {  
 **this**.**ocrRate** = **this**.strSimilarity2Percent(***JSON***.stringify(newValue), ***JSON***.stringify(**this**.**historyData**)) \* 100;  
 *// this.ocrRate = this.strSimilarity2Percent('ab2cdefgfgf', 'ab2cde') \* 100;* },  
 **deep**: **true** }  
 },  
 created() {  
  
 }  
};  
</**script**>

<**style lang="less" scoped**>  
 .**page-box**{  
 **padding**: 10**px**;  
 **position**: **relative**;  
 **box-sizing**: **border-box**;  
 **overflow**: **hidden**;  
 }  
</**style**>  
<**style lang="less"**>  
 .**el-table** .**warning-row** {  
 **background**: **lightcoral**;  
 }  
 .**el-table** .**yellow-row** {  
 **background**: **lemonchiffon**;  
 }  
 .**vgm** .**ivu-modal-body**{  
 **padding**: 18**px !important**;  
 }  
</**style**>  
<**template**>  
 <**div class="page-box"**>  
 <**search-modal :show.sync="searchMenu" :width="**370**" modal-height="calc(100vh - 90px)"**>  
 <**Row class="mb-10"**>  
 <**i-col span="10"**>  
 <**search-modal-title :width="**100**"**><**i class="iconfont icon-sousuo"**></**i**>{{this.$t('common.advanceSearch')}}</**search-modal-title**>  
 </**i-col**>  
 <**i-col span="14" style="text-align: right;"**>  
 <**Button size="small" type="primary" @click="**handleQuery**"**>{{this.$t('common.search')}}</**Button**>  
 <**Button size="small" type="primary" @click="**handleReset**"**>{{this.$t('common.reset')}}</**Button**>  
 </**i-col**>  
 </**Row**>  
 <**Row class="mb-10 search-modal-box"**>  
 <**i-col**>  
 <**Form :model="searchForm" :label-width="**100**"**>  
  
 <**FormItem :label="**this.$t('OcrField.id')**"**>  
 <**Input size="small" v-model="searchForm**.**id" clearable**></**Input**>  
 </**FormItem**>  
  
 <**FormItem :label="**this.$t('OcrField.sheetType')**"**>  
 <**Input size="small" v-model="searchForm**.**sheetType" clearable**></**Input**>  
 </**FormItem**>  
  
 <**FormItem :label="**this.$t('OcrField.fieldCode')**"**>  
 <**Input size="small" v-model="searchForm**.**fieldCode" clearable**></**Input**>  
 </**FormItem**>  
  
 <**FormItem :label="**this.$t('OcrField.fieldName')**"**>  
 <**Input size="small" v-model="searchForm**.**fieldName" clearable**></**Input**>  
 </**FormItem**>  
  
 <**FormItem :label="**this.$t('OcrField.preType')**"**>  
 <**Input size="small" v-model="searchForm**.**preType" clearable**></**Input**>  
 </**FormItem**>  
  
 <**FormItem :label="**this.$t('OcrField.preStrategy')**"**>  
 <**Input size="small" v-model="searchForm**.**preStrategy" clearable**></**Input**>  
 </**FormItem**>  
  
 <**FormItem :label="**this.$t('OcrField.recogType')**"**>  
 <**Input size="small" v-model="searchForm**.**recogType" clearable**></**Input**>  
 </**FormItem**>  
  
 <**FormItem :label="**this.$t('OcrField.recogStrategy')**"**>  
 <**Input size="small" v-model="searchForm**.**recogStrategy" clearable**></**Input**>  
 </**FormItem**>  
  
 <**FormItem :label="**this.$t('OcrField.directSearchFlag')**"**>  
 <**Input size="small" v-model="searchForm**.**directSearchFlag" clearable**></**Input**>  
 </**FormItem**>  
  
 <**FormItem :label="**this.$t('OcrField.directSearchReg')**"**>  
 <**Input size="small" v-model="searchForm**.**directSearchReg" clearable**></**Input**>  
 </**FormItem**>  
  
 <**FormItem :label="**this.$t('OcrField.emptyFlag')**"**>  
 <**Input size="small" v-model="searchForm**.**emptyFlag" clearable**></**Input**>  
 </**FormItem**>  
  
 <**FormItem :label="**this.$t('OcrField.validFlag')**"**>  
 <**Input size="small" v-model="searchForm**.**validFlag" clearable**></**Input**>  
 </**FormItem**>  
  
 <**FormItem :label="**this.$t('OcrField.remark')**"**>  
 <**Input size="small" v-model="searchForm**.**remark" clearable**></**Input**>  
 </**FormItem**>  
  
 <**FormItem :label="**this.$t('OcrField.createdBy')**"**>  
 <**Input size="small" v-model="searchForm**.**createdBy" clearable**></**Input**>  
 </**FormItem**>  
  
 <**FormItem :label="**this.$t('OcrField.createdTime')**"**>  
 <**Input size="small" v-model="searchForm**.**createdTime" clearable**></**Input**>  
 </**FormItem**>  
  
 <**FormItem :label="**this.$t('OcrField.lastModifiedBy')**"**>  
 <**Input size="small" v-model="searchForm**.**lastModifiedBy" clearable**></**Input**>  
 </**FormItem**>  
  
 <**FormItem :label="**this.$t('OcrField.lastModifiedTime')**"**>  
 <**Input size="small" v-model="searchForm**.**lastModifiedTime" clearable**></**Input**>  
 </**FormItem**>  
  
 <**FormItem :label="**this.$t('OcrField.deleteFlag')**"**>  
 <**Input size="small" v-model="searchForm**.**deleteFlag" clearable**></**Input**>  
 </**FormItem**>  
  
 </**Form**>  
 </**i-col**>  
 </**Row**>  
 </**search-modal**>  
 <**Card**>  
 <**Row class="mb-10"**>  
 <**i-col span="20"**>  
 <**menu-content-layout-title**>{{this.$t('OcrField.title')}}</**menu-content-layout-title**>  
 <**ButtonGroup**>  
 <**Button size="small" type="primary" @click="**addOne**"**>{{this.$t('common.new')}}</**Button**>  
 <**Button size="small" type="ghost" @click="**openOne**" :disabled="selectRows**.**length** === 0**"**>{{this.$t('common.open')}}</**Button**>  
 <**Button size="small" type="dashed" @click="**deleteConfirm**" :disabled="selectRows**.**length** === 0**"**>{{this.$t('common.delete')}}</**Button**>  
 </**ButtonGroup**>  
 </**i-col**>  
 <**i-col span="4" style="text-align: right;"**>  
 <**Button size="small" type="ghost" icon="refresh" @click="**initGridData**"**></**Button**>  
 <**Button size="small" type="ghost" icon="ios-search" @click="searchMenu**=!**searchMenu"**>{{this.$t('common.advanceSearch')}}</**Button**>  
 </**i-col**>  
 </**Row**>  
 <**el-table  
 :data="tableData**.**list"  
 height="calc(100vh - 180px)"  
 border  
 style="width: 100%"  
 size="mini"  
 highlight-current-row  
 @row-contextmenu="**onRowContentMenu**"  
 @row-click="**onRowClick**"  
 @row-dblclick="**dblClickOpen**"  
 @select="**selectRow**"  
 @select-all="**selectAllRows**"  
 @selection-change="**onSelectionChange**"  
 oncontextmenu="return false**;**"  
 ref="multipleTable"  
 sortable="custom"  
 @sort-change="**onTableSort**"  
 :row-class-name="**tableRowclassCode**"  
 v-loading="tableData**.**columnsLoading** || **tableData**.**dataLoading"** >  
 <**el-table-column  
 type="selection"  
 width="36"**>  
 </**el-table-column**>  
 <**el-table-column  
 type="index"  
 align="center"  
 width="36"**>  
 </**el-table-column**>  
 <**el-table-column  
 v-for="**(***item***, ***key***) in **tableData**.**columns" :key="*key*"  
 v-if="*item***.**visible"  
 :prop="*item***.**key"  
 :label="*item***.**title"  
 :width="*item***.**width"  
 :align="*item***.**align** || ''**"  
 :sortable="*item***.**sortable**!==**undefined**?'custom': false**"  
 :show-overflow-tooltip="*item***.**type** !== 'fieldStatus' && ***item***.**type** !== 'allowStatus'**"** >  
 <**template slot-scope="scope"**>  
 <**template v-if="*item***.**type** !== 'template'**"**>  
 {{**$util**.tableRowFormat(**scope**.row, ***item***)}}  
 </**template**>  
 <**template v-else**>  
 <**table-cell-render :row="scope**.row**" :option="*item*"**></**table-cell-render**>  
 </**template**>  
 </**template**>  
 </**el-table-column**>  
 </**el-table**>  
 <**context-menu ref="tableContextMenu"**>  
 <**li class="ctx-item" @click="**onEdit()**"**>{{this.$t('common.open')}}</**li**>  
 <**li class="ctx-item" @click="**onDel()**"**>{{this.$t('common.delete')}}</**li**>  
 </**context-menu**>  
 <**Page class="mt-10" :total="tableData**.**total" :page-size="tableData**.**size" :page-size-opts="tableData**.**pageOptionsList" @on-change="**onGridPageChange**" @on-page-size-change="**onGridPageSizeChange**" size="small" show-elevator show-sizer placement="top"**></**Page**>  
 <**table-options name="OcrFieldList" :data.sync="tableData**.**columns" :base-data="tableData**.**defaultColumns" :column-width-obj="tableData**.**columnWidthObj"**></**table-options**>  
 </**Card**>  
 </**div**>  
</**template**>  
<**script**>  
**import** api **from '@/api/ocr/ocrField'**;  
**import** tableApi **from '@/api/setting/tableColumns'**;  
**import** util **from '@/libs/util'**;  
**import** TableOptions **from '@/components/table/table-options/index'**;  
  
**export default** {  
 **name**: **'OcrField'**,  
 **components**: {  
 TableOptions  
 },  
  
 data () {  
 **return** {  
 **tableData**: {  
 **columnsLoading**: **true**,  
 **dataLoading**: **true**,  
 **pageNum**: 1,  
 **size**: **this**.$store.**state**.**sysSettings**.**pageSize**,  
 **pageOptionsList**: **this**.$store.**state**.**sysSettings**.**pageOptionsList**,  
 **total**: 0,  
 **sortList**: [],  
 **filterList**: [],  
 **columns**: [{**title**: **''**, **key**: **'none'**, **visible**: **true**}],  
 **id**: 0,  
 **classCode**: **''**,  
 **schemaName**: **''**,  
 **defaultColumns**: [  
  
 {  
 **title**: **this**.$t(**'OcrField.id'**),  
 **key**: **'id'**,  
 **width**: 100,  
 **sortable**: **true**,  
 **visible**: **true** },  
  
 {  
 **title**: **this**.$t(**'OcrField.sheetType'**),  
 **key**: **'sheetType'**,  
 **width**: 100,  
 **sortable**: **true**,  
 **visible**: **true** },  
  
 {  
 **title**: **this**.$t(**'OcrField.fieldCode'**),  
 **key**: **'fieldCode'**,  
 **width**: 100,  
 **sortable**: **true**,  
 **visible**: **true** },  
  
 {  
 **title**: **this**.$t(**'OcrField.fieldName'**),  
 **key**: **'fieldName'**,  
 **width**: 100,  
 **sortable**: **true**,  
 **visible**: **true** },  
  
 {  
 **title**: **this**.$t(**'OcrField.preType'**),  
 **key**: **'preType'**,  
 **width**: 100,  
 **sortable**: **true**,  
 **visible**: **true** },  
  
 {  
 **title**: **this**.$t(**'OcrField.preStrategy'**),  
 **key**: **'preStrategy'**,  
 **width**: 100,  
 **sortable**: **true**,  
 **visible**: **true** },  
  
 {  
 **title**: **this**.$t(**'OcrField.recogType'**),  
 **key**: **'recogType'**,  
 **width**: 100,  
 **sortable**: **true**,  
 **visible**: **true** },  
  
 {  
 **title**: **this**.$t(**'OcrField.recogStrategy'**),  
 **key**: **'recogStrategy'**,  
 **width**: 100,  
 **sortable**: **true**,  
 **visible**: **true** },  
  
 {  
 **title**: **this**.$t(**'OcrField.directSearchFlag'**),  
 **key**: **'directSearchFlag'**,  
 **width**: 100,  
 **sortable**: **true**,  
 **visible**: **true** },  
  
 {  
 **title**: **this**.$t(**'OcrField.directSearchReg'**),  
 **key**: **'directSearchReg'**,  
 **width**: 100,  
 **sortable**: **true**,  
 **visible**: **true** },  
  
 {  
 **title**: **this**.$t(**'OcrField.emptyFlag'**),  
 **key**: **'emptyFlag'**,  
 **width**: 100,  
 **sortable**: **true**,  
 **visible**: **true** },  
  
 {  
 **title**: **this**.$t(**'OcrField.validFlag'**),  
 **key**: **'validFlag'**,  
 **width**: 100,  
 **sortable**: **true**,  
 **visible**: **true** },  
  
 {  
 **title**: **this**.$t(**'OcrField.parentId'**),  
 **key**: **'parentId'**,  
 **width**: 100,  
 **sortable**: **true**,  
 **visible**: **true** },  
  
 {  
 **title**: **this**.$t(**'OcrField.fieldStructureType'**),  
 **key**: **'fieldStructureType'**,  
 **width**: 100,  
 **sortable**: **true**,  
 **visible**: **true** },  
  
 {  
 **title**: **this**.$t(**'OcrField.remark'**),  
 **key**: **'remark'**,  
 **width**: 100,  
 **sortable**: **true**,  
 **visible**: **true** },  
  
 {  
 **title**: **this**.$t(**'OcrField.createdBy'**),  
 **key**: **'createdBy'**,  
 **width**: 100,  
 **sortable**: **true**,  
 **visible**: **true** },  
  
 {  
 **title**: **this**.$t(**'OcrField.createdTime'**),  
 **key**: **'createdTime'**,  
 **width**: 100,  
 **sortable**: **true**,  
 **visible**: **true** },  
  
 {  
 **title**: **this**.$t(**'OcrField.lastModifiedBy'**),  
 **key**: **'lastModifiedBy'**,  
 **width**: 100,  
 **sortable**: **true**,  
 **visible**: **true** },  
  
 {  
 **title**: **this**.$t(**'OcrField.lastModifiedTime'**),  
 **key**: **'lastModifiedTime'**,  
 **width**: 100,  
 **sortable**: **true**,  
 **visible**: **true** },  
  
 {  
 **title**: **this**.$t(**'OcrField.deleteFlag'**),  
 **key**: **'deleteFlag'**,  
 **width**: 100,  
 **sortable**: **true**,  
 **visible**: **true** }  
  
 ],  
 **columnWidthObj**: **null**,  
 **list**: []  
 },  
 **searchForm**: {  
  
 **id**: **''**,  
 **sheetType**: **''**,  
 **fieldCode**: **''**,  
 **fieldName**: **''**,  
 **preType**: **''**,  
 **preStrategy**: **''**,  
 **recogType**: **''**,  
 **recogStrategy**: **''**,  
 **directSearchFlag**: **''**,  
 **directSearchReg**: **''**,  
 **emptyFlag**: **''**,  
 **validFlag**: **''**,  
 **remark**: **''**,  
 **createdBy**: **''**,  
 **createdTime**: **''**,  
 **lastModifiedBy**: **''**,  
 **lastModifiedTime**: **''**,  
 **deleteFlag**: **''** },  
 **selectRows**: [],  
 **rightClickRow**: **null**,  
 **searchMenu**: **false** };  
 },  
 **methods**: {  
 *// 新增和打开方法* addOne: **function** (type) {  
 **this**.$router.**push**({  
 **name**: **'OcrFieldEdit'**,  
 **params**: {**op**: **'refresh'**}  
 });  
 },  
 openOne: **function** (type) {  
 **if** (**this**.**selectRows**.**length** <= 0) {  
 **this**.$Message.warning(**'请选择一条记录'**);  
 **return false**;  
 }  
 **if** (**this**.**selectRows**.**length** > 1) {  
 **this**.$Message.warning(**'请确定一条记录'**);  
 **return false**;  
 }  
 **this**.$router.**push**({  
 **name**: **'OcrFieldEdit'**,  
 **query**: {**id**: **this**.**selectRows**[0].**id**},  
 **params**: {**op**: **'refresh'**}  
 });  
 },  
 *// 删除和批量删除* deleteConfirm () {  
 **if** (**this**.**selectRows**.**length** <= 0) {  
 **this**.$Message.warning(**this**.$t(**'common.pleaseSelectDeleteRecord'**));  
 **return false**;  
 }  
 **let** content = **'<p>将要删除的数据有:</p>'**;  
 **this**.**selectRows**.forEach(**function** (value) {  
 content += **'<p>'** + value.**id** + **'</p>'**;  
 });  
 **this**.$Modal.confirm({  
 **title**: **this**.$t(**'common.deleteConfirmTitle'**),  
 **content**: content,  
 onOk: () => {  
 **this**.deleteItemInfo();  
 }  
 });  
 },  
 deleteItemInfo() {  
 **if** (**this**.**selectRows**.**length** <= 0) {  
 **this**.$Message.warning(**this**.$t(**'common.pleaseSelectDeleteRecord'**));  
 **return false**;  
 }  
 **let** ids = [];  
 **for** (**let** idx **in this**.**selectRows**) {  
 ids.push(**this**.**selectRows**[idx].**id**);  
 }  
 api.OcrFieldEditDelete(ids).then(response => {  
 **if** (response.**data**.returnCode === 200) {  
 **this**.initGridData();  
 **this**.$Message.success(**this**.$t(response.**data**.returnMsg));  
 }  
 }).catch(error => {  
 ***console***.log(**this**.$t(**'common.systemError'**), error);  
 });  
 },  
  
 handleReset() {  
 **this**.**searchForm** = {  
 **id**: **''**,  
 **sheetType**: **''**,  
 **fieldCode**: **''**,  
 **fieldName**: **''**,  
 **preType**: **''**,  
 **preStrategy**: **''**,  
 **recogType**: **''**,  
 **recogStrategy**: **''**,  
 **directSearchFlag**: **''**,  
 **directSearchReg**: **''**,  
 **emptyFlag**: **''**,  
 **validFlag**: **''**,  
 **parentId**: **''**,  
 **fieldStructureType**: **''**,  
 **remark**: **''**,  
 **createdBy**: **''**,  
 **createdTime**: **''**,  
 **lastModifiedBy**: **''**,  
 **lastModifiedTime**: **''**,  
 **deleteFlag**: **''** };  
 **let** filterList = util.searchFormFormat(**this**.**searchForm**);  
 **this**.**tableData**.**filterList** = filterList;  
 **this**.initGridData();  
 },  
 handleQuery() {  
 **this**.**listLoading** = **true**;  
 **let** filterList = util.searchFormFormat(**this**.**searchForm**);  
 **this**.**tableData**.**filterList** = filterList;  
 **this**.initGridData();  
 },  
 formatter (row, column) {  
 **return** row.**address**;  
 },  
 filterTag (value, row) {  
 **return** row.**tag** === value;  
 },  
 filterHandler (value, row, column) {  
 **const** property = column[**'property'**];  
 **return** row[property] === value;  
 },  
 onRowContentMenu (row, event) {  
 **this**.**rightClickRow** = row;  
 **this**.$refs.multipleTable.clearSelection();  
 **this**.$refs.multipleTable.toggleRowSelection(row);  
 **this**.**selectRows**.push(row);  
 **this**.$refs.tableContextMenu.open(event);  
 },  
 selectRow(selection) {  
 **this**.**selectRows** = selection;  
 ***console***.log(**this**.**selectRows**);  
 },  
 onSelectionChange (selection) {  
 **this**.**selectRows** = selection;  
 ***console***.log(**this**.**selectRows**);  
 },  
 selectAllRows(selection) {  
 **this**.**selectRows** = selection;  
 ***console***.log(**this**.**selectRows**);  
 },  
 onRowClick (row, event, column) {  
 **this**.$refs.multipleTable.toggleRowSelection(row);  
 },  
 *// 双击打开一条记录* dblClickOpen(row) {  
 **this**.$refs.multipleTable.clearSelection();  
 **this**.$refs.multipleTable.toggleRowSelection(row);  
 **this**.**selectRows** = [];  
 **this**.**selectRows**.push(row);  
 **this**.$router.**push**({  
 **name**: **'OcrFieldEdit'**,  
 **query**: { **id**: row.**id** },  
 **params**: {**op**: **'refresh'**}  
 });  
 },  
 onEdit () {  
 **this**.$router.**push**({  
 **name**: **'OcrFieldEdit'**,  
 **query**: { **id**: **this**.**rightClickRow**.**id** },  
 **params**: {**op**: **'refresh'**}  
 });  
 },  
 onDel () {  
 **const** idList = [**this**.**rightClickRow**.**id**];  
 api.OcrFieldEditDelete(idList).then(({data}) => {  
 **if** (data.returnCode === 200) {  
 **this**.initGridData();  
 } **else** {  
 **this**.$Message.warning(**'数据异常,请重试！'**);  
 }  
 });  
 },  
  
 tableRowclassCode() {  
 **return ''**;  
 },  
  
 getLayoutSettings () {  
 **this**.**tableData**.**columnsLoading** = **true**;  
 tableApi.getLayoutSettings({**tableName**: **'OcrFieldList'**}).then(({data}) => {  
 **if** (data.returnCode === 200) {  
 **if** (data.returnData.**length** > 0) {  
 **this**.**tableData**.**columns** = ***JSON***.parse(data.returnData[0].**columnsSettings**);  
 } **else** {  
 **this**.**tableData**.**columns** = **this**.**tableData**.**defaultColumns**;  
 }  
 } **else** {  
 **this**.**tableData**.**columns** = **this**.**tableData**.**defaultColumns**;  
 */\* this.$Message.warning('数据异常,请重试！'); \*/* }  
 **this**.$nextTick(() => {  
 **this**.**tableData**.**columnsLoading** = **false**;  
 });  
 });  
 },  
  
 initGridData () {  
 **this**.**tableData**.**dataLoading** = **true**;  
 **this**.**selectRows** = [];  
  
 **let** body = {  
 **pageNum**: **this**.**tableData**.**pageNum**,  
 **pageSize**: **this**.**tableData**.**size**,  
 **filterList**: **this**.**tableData**.**filterList**,  
 **sortList**: **this**.**tableData**.**sortList** };  
 api.getOcrFieldList(body).then(({data}) => {  
 ***console***.log(data);  
 **if** (data.returnCode === 200) {  
 **this**.**tableData**.**list** = data.returnData.pageInfo.**list**;  
 **this**.**tableData**.**total** = data.returnData.pageInfo.**total**;  
 } **else** {  
 **this**.$Message.warning(**'数据异常,请重试！'**);  
 }  
 **this**.$nextTick(() => {  
 **this**.**tableData**.**dataLoading** = **false**;  
 });  
 });  
 },  
 onGridPageChange (pageNum) {  
 **this**.**tableData**.**pageNum** = pageNum;  
 **this**.initGridData();  
 },  
 onGridPageSizeChange (size) {  
 **this**.**tableData**.**pageNum** = 1;  
 **this**.**tableData**.**size** = size;  
 **this**.initGridData();  
 },  
 onTableSort (result) {  
 **if** (result.**order** === **'descending'**) {  
 **let** sortItem = {  
 **sortKey**: result.prop,  
 **sortValue**: **'desc'** };  
 **this**.**tableData**.**sortList** = [sortItem];  
 } **else if** (result.**order** === **'ascending'**) {  
 **let** sortItem = {  
 **sortKey**: result.prop,  
 **sortValue**: **'asc'** };  
 **this**.**tableData**.**sortList** = [sortItem];  
 } **else** {  
 **this**.**tableData**.**sortList** = [];  
 }  
 **this**.initGridData();  
 },  
 getAllBusinessType () {  
 api.getAllBusinessType().then(({data}) => {  
 **if** (data.returnCode === 200) {  
 *// this.goodsTypeList = data.returnData.goodsTypeList;* } **else** {  
 **this**.$Message.warning(data.returnMsg);  
 }  
 });  
 }  
 },  
 activated () {  
 **if** (**this**.$route.**params**.**op** !== **undefined** && **this**.$route.**params**.**op** === **'refresh'**) {  
 *// 刷新数据* **this**.initGridData();  
 }  
 },  
  
 mounted() {  
 **eventBus**.$on(**this**.$route.**name** + **'\_refresh'**, (body) => {  
 **this**.initGridData();  
 });  
 },  
 created () {  
 **this**.getLayoutSettings();  
 **this**.initGridData();  
 }  
};  
</**script**>

<**style lang="less"**>  
 .**ivu-form-item-error-tip**{  
 **z-index**: 999;  
 **padding-top**: 0**px**;  
 }  
</**style**>  
<**template**>  
 <**default-page-box**>  
 <**template slot="tools-bar"**>  
 <**Badge style="margin-right: 5px" dot v-if="modifyFlag"**>  
 <**Button :disabled="canEdit" type="info" size="small" @click="**save**" :loading="saveLoading"**>保存</**Button**>  
 </**Badge**>  
 <**Button style="margin-right: 5px" :disabled="canEdit" v-if="**!**modifyFlag" type="info" size="small" @click="**save**" :loading="saveLoading"**>保存</**Button**>  
 <**Button style="margin-right: 5px" type="ghost" size="small" @click="**back**"**>返回</**Button**>  
 </**template**>  
 <**template slot="content"**>  
 <**Card :style="**{**height**: 'calc(100vh - 125px)'}**"**>  
 <**Form :label-width="**100**" :model="OcrField" ref="OcrFieldForm" :rules="ruleValidate"**>  
 <**Row**>  
  
 <**i-col span="12"**>  
 <**FormItem :label="**this.$t('OcrField.sheetType')**" prop="sheetType"**>  
 <**Input size="small" v-model="OcrField**.**sheetType" clearable**></**Input**>  
 </**FormItem**>  
 </**i-col**>  
  
 <**i-col span="12"**>  
 <**FormItem :label="**this.$t('OcrField.fieldCode')**" prop="fieldCode"**>  
 <**Input size="small" v-model="OcrField**.**fieldCode" clearable**></**Input**>  
 </**FormItem**>  
 </**i-col**>  
  
 <**i-col span="12"**>  
 <**FormItem :label="**this.$t('OcrField.fieldName')**" prop="fieldName"**>  
 <**Input size="small" v-model="OcrField**.**fieldName" clearable**></**Input**>  
 </**FormItem**>  
 </**i-col**>  
  
 <**i-col span="12"**>  
 <**FormItem :label="**this.$t('OcrField.preType')**" prop="preType"**>  
 <**Input size="small" v-model="OcrField**.**preType" clearable**></**Input**>  
 </**FormItem**>  
 </**i-col**>  
  
 <**i-col span="12"**>  
 <**FormItem :label="**this.$t('OcrField.preStrategy')**" prop="preStrategy"**>  
 <**Input size="small" v-model="OcrField**.**preStrategy" clearable**></**Input**>  
 </**FormItem**>  
 </**i-col**>  
  
 <**i-col span="12"**>  
 <**FormItem :label="**this.$t('OcrField.recogType')**" prop="recogType"**>  
 <**Input size="small" v-model="OcrField**.**recogType" clearable**></**Input**>  
 </**FormItem**>  
 </**i-col**>  
  
 <**i-col span="12"**>  
 <**FormItem :label="**this.$t('OcrField.recogStrategy')**" prop="recogStrategy"**>  
 <**Input size="small" v-model="OcrField**.**recogStrategy" clearable**></**Input**>  
 </**FormItem**>  
 </**i-col**>  
  
 *<!--<i-col span="8">  
 <FormItem :label="this.$t('OcrField.directSearchFlag')" prop="directSearchFlag">  
 <Input size="small" v-model="OcrField.directSearchFlag" clearable></Input>  
 </FormItem>  
 </i-col>-->* <**i-col span="12"**>  
 <**FormItem :label="**this.$t('OcrField.directSearchFlag')**" prop="validFlag"**>  
 <**i-switch v-model="OcrField**.**directSearchFlag" size="large"**>  
 <**span slot="open"**>是</**span**>  
 <**span slot="close"**>否</**span**>  
 </**i-switch**>  
 </**FormItem**>  
 </**i-col**>  
  
 <**i-col span="12"**>  
 <**FormItem :label="**this.$t('OcrField.directSearchReg')**" prop="directSearchReg"**>  
 <**Input size="small" v-model="OcrField**.**directSearchReg" clearable**></**Input**>  
 </**FormItem**>  
 </**i-col**>  
  
 *<!--<i-col span="8">  
 <FormItem :label="this.$t('OcrField.emptyFlag')" prop="emptyFlag">  
 <Input size="small" v-model="OcrField.emptyFlag" clearable></Input>  
 </FormItem>  
 </i-col>-->* <**i-col span="12"**>  
 <**FormItem :label="**this.$t('OcrField.emptyFlag')**" prop="validFlag"**>  
 <**i-switch v-model="OcrField**.**emptyFlag" size="large"**>  
 <**span slot="open"**>是</**span**>  
 <**span slot="close"**>否</**span**>  
 </**i-switch**>  
 </**FormItem**>  
 </**i-col**>  
  
 *<!--<i-col span="8">  
 <FormItem :label="this.$t('OcrField.validFlag')" prop="validFlag">  
 <Input size="small" v-model="OcrField.validFlag" clearable></Input>  
 </FormItem>  
 </i-col>-->* <**i-col span="12"**>  
 <**FormItem :label="**this.$t('OcrField.validFlag')**" prop="validFlag"**>  
 <**i-switch v-model="OcrField**.**validFlag" size="large"**>  
 <**span slot="open"**>是</**span**>  
 <**span slot="close"**>否</**span**>  
 </**i-switch**>  
 </**FormItem**>  
 </**i-col**>  
  
 <**i-col span="12"**>  
 <**FormItem :label="**this.$t('OcrField.parentId')**" prop="parentId"**>  
 <**Input size="small" v-model="OcrField**.**parentId" clearable**></**Input**>  
 </**FormItem**>  
 </**i-col**>  
  
 <**i-col span="12"**>  
 <**FormItem :label="**this.$t('OcrField.fieldStructureType')**" prop="fieldStructureType"**>  
 *<!--<Input size="small" v-model="OcrField.fieldStructureType" clearable></Input>-->* <**Select v-model="OcrField**.**fieldStructureType" size="small"**>  
 <**Option :value="**0**" :key="**0**"**>默认类型</**Option**>  
 <**Option :value="**1**" :key="**1**"**>表格类型</**Option**>  
 </**Select**>  
 </**FormItem**>  
 </**i-col**>  
 *<!--<i-col span="8">  
 <FormItem :label="this.$t('OcrField.remark')" prop="remark">  
 <Input size="small" v-model="OcrField.remark" clearable></Input>  
 </FormItem>  
 </i-col>-->* <**i-col span="24"**>  
 <**FormItem :label="**this.$t('OcrField.remark')**" prop="remark"**>  
 <**Input size="small" v-model="OcrField**.**remark" type="textarea" :autosize="**{**minRows**: 3, **maxRows**: 5}**"**></**Input**>  
 </**FormItem**>  
 </**i-col**>  
  
 *<!--<i-col span="8">  
 <FormItem :label="this.$t('OcrField.createdBy')" prop="createdBy">  
 <Input size="small" v-model="OcrField.createdBy" clearable></Input>  
 </FormItem>  
 </i-col>  
  
 <i-col span="8">  
 <FormItem :label="this.$t('OcrField.createdTime')" prop="createdTime">  
 <Input size="small" v-model="OcrField.createdTime" clearable></Input>  
 </FormItem>  
 </i-col>  
  
 <i-col span="8">  
 <FormItem :label="this.$t('OcrField.lastModifiedBy')" prop="lastModifiedBy">  
 <Input size="small" v-model="OcrField.lastModifiedBy" clearable></Input>  
 </FormItem>  
 </i-col>  
  
 <i-col span="8">  
 <FormItem :label="this.$t('OcrField.lastModifiedTime')" prop="lastModifiedTime">  
 <Input size="small" v-model="OcrField.lastModifiedTime" clearable></Input>  
 </FormItem>  
 </i-col>  
  
 </FormItem>  
 </i-col>-->* </**Row**>  
 </**Form**>  
 </**Card**>  
 </**template**>  
 </**default-page-box**>  
</**template**>  
  
<**script**>  
**import** api **from '@/api/ocr/ocrField'**;  
**import** apiBusinessCode **from '@/api/codec/business\_code'**;  
**import** DefaultPageBox **from '@/components/page/default-page-box/index'**;  
**import** \_ **from 'lodash'**;  
  
**export default** {  
 **components**: {  
 DefaultPageBox  
 },  
 **name**: **'OcrFieldEdit'**,  
 **props**: {  
 **id**: {  
 **type**: ***String***,  
 **default**: **''** }  
 },  
 data() {  
 **return** {  
 **canEdit**: **false**,  
 **saveFlag**: **false**,  
 **modifyFlag**: **''**,  
 **saveLoading**: **false**,  
  
 **OcrFieldOrig**: {  
 **sheetType**: **''**,  
 **fieldCode**: **''**,  
 **fieldName**: **''**,  
 **preType**: **''**,  
 **preStrategy**: **''**,  
 **recogType**: **''**,  
 **recogStrategy**: **''**,  
 **directSearchFlag**: **false**,  
 **directSearchReg**: **''**,  
 **emptyFlag**: **false**,  
 **validFlag**: **false**,  
 **fieldStructureType**: **''**,  
 **parentId**: **''**,  
 **remark**: **''**,  
 **createdBy**: **''**,  
 **createdTime**: **''**,  
 **lastModifiedBy**: **''**,  
 **lastModifiedTime**: **''**,  
 **deleteFlag**: **''** },  
  
 **OcrField**: {  
 **sheetType**: **''**,  
 **fieldCode**: **''**,  
 **fieldName**: **''**,  
 **preType**: **''**,  
 **preStrategy**: **''**,  
 **recogType**: **''**,  
 **recogStrategy**: **''**,  
 **directSearchFlag**: **false**,  
 **directSearchReg**: **''**,  
 **emptyFlag**: **false**,  
 **validFlag**: **false**,  
 **fieldStructureType**: **''**,  
 **parentId**: **''**,  
 **remark**: **''**,  
 **createdBy**: **''**,  
 **createdTime**: **''**,  
 **lastModifiedBy**: **''**,  
 **lastModifiedTime**: **''**,  
 **deleteFlag**: **''** },  
  
 **tableData**: {  
 **loading**: **false**,  
 **list**: [],  
 **total**: 0  
 },  
 **listQuery**: {  
 **pageNum**: 1,  
 **pageSize**: 10,  
 **filterList**: [],  
 **sortList**: [],  
 **searchKey**: **''** },  
  
 **ruleValidate**: {  
  
 */\* id: [  
 { required: true, message: 'id必填', trigger: 'change' }  
 ], \*/* **sheetType**: [  
 { **required**: **true**, **message**: **'sheetType必填'**, **trigger**: **'change'** }  
 ],  
 **fieldCode**: [  
 { **required**: **true**, **message**: **'fieldCode必填'**, **trigger**: **'change'** }  
 ],  
 **fieldName**: [  
 { **required**: **true**, **message**: **'fieldName必填'**, **trigger**: **'change'** }  
 ]  
 }  
 };  
 },  
 **methods**: {  
 numberWithCommas(x) {  
 **if** (x === **undefined** || x === **null** || x === 0) **return '0'**;  
 **let** parts = x.toString().split(**'.'**);  
 **if** (parts.**length** === 2) {  
 parts[1] = parts[1].substring(0, 2);  
 }  
 parts[0] = parts[0].replace(/\B(?=(\d{3})+(?!\d))/g, **','**);  
 **return** parts.join(**'.'**);  
 },  
  
 *// 保存* save() { *// 整个页面保存或修改* **this**.$refs[**'OcrFieldForm'**].validate((valid) => {  
 **if** (!valid) {  
 **return false**;  
 } **else** {  
 **let** body = **this**.**OcrField**;  
 **this**.**saveLoading** = **true**; *// submitLoading* **if** (body.**id** > 0) {  
 **this**.updateEntity(body);  
 } **else** {  
 **this**.createEntity(body);  
 }  
 }  
 });  
 },  
  
 updateEntity(body) {  
 **if** (body.**id** > 0) {  
 *// 修改* api.updateOcrField(body).then(response => {  
 **const** status = response.**data**.returnCode;  
 **const** message = response.**data**.returnMsg;  
 **if** (status === 200) {  
 **this**.$notify({  
 **title**: **this**.$t(**'common.success'**),  
 **message**: **this**.$t(message),  
 **type**: **'success'**,  
 **duration**: 2000  
 });  
 **this**.getWholeInfo(body.**id**);  
 } **else** {  
 **this**.**$util**.modalError({  
 **title**: **this**.$t(**'common.failure'**),  
 **content**: **this**.$t(message)  
 });  
 }  
 **this**.**saveLoading** = **false**;  
 }).catch(error => {  
 **this**.**saveLoading** = **false**;  
 **this**.$Message.success(**'保存发生错误'**, error);  
 });  
 **return false**;  
 }  
 },  
  
 createEntity(body) {  
 api.createOcrField(body).then(response => {  
 **const** status = response.**data**.returnCode;  
 **const** message = response.**data**.returnMsg;  
 **const** resultData = response.**data**.returnData;  
 **if** (status === 200) {  
 **this**.$notify({  
 **title**: **this**.$t(**'common.success'**),  
 **message**: **this**.$t(message),  
 **type**: **'success'**,  
 **duration**: 2000  
 });  
 **this**.getWholeInfo(resultData, **true**);  
 } **else** {  
 **this**.**$util**.modalError({  
 **title**: **this**.$t(**'common.failure'**),  
 **content**: **this**.$t(message)  
 });  
 }  
 **this**.**saveLoading** = **false**;  
 }).catch(error => {  
 **this**.**saveLoading** = **false**;  
 **this**.$Message.success(**'保存发生错误'**, error);  
 });  
 },  
 *// 返回上一页* back() {  
 **this**.$router.**push**({**name**: **'OcrField'**});  
 },  
  
 getWholeInfo(id, getReason) {  
 *// 获取全部信息* api.getOcrField(id).then(response => {  
 **if** (response.**data**.returnCode === 200) {  
 **this**.**OcrField** = response.**data**.returnData;  
 } **else** {  
 **const** message = response.**data**.returnMsg;  
 **this**.$notify({  
 **title**: **this**.$t(**'common.failure'**),  
 **message**: **this**.$t(message),  
 **type**: **'warning'**,  
 **duration**: 2000  
 });  
 }  
 }).catch(error => {  
 ***console***.log(**'获取信息发生错误'**, error);  
 });  
 }  
 },  
 **watch**: {  
 **OcrField**: {  
 handler(newValue) {  
 **if** (**this**.**historyData** === **null**) {  
 **return false**;  
 }  
 *// console.log(newValue,this.historyData.OcrField);* **this**.**modifyFlag** = !(\_.isEqual(newValue, **this**.**historyData**));  
 },  
 **deep**: **true** }  
 },  
 activated () {  
 **if** (**this**.$route.**query**.**id** !== **undefined** && **this**.$route.**query**.**id** !== **null**) {  
 **this**.getWholeInfo(**this**.$route.**query**.**id**);  
 } **else** {  
 **this**.**OcrField** = **this**.**OcrFieldOrig**;  
 }  
 },  
  
 created() {  
 **if** (**this**.$route.**query**.**id** !== **undefined** && **this**.$route.**query**.**id** !== **null**) {  
 **this**.getWholeInfo(**this**.$route.**query**.**id**);  
 } **else** {  
 **this**.**OcrField** = **this**.**OcrFieldOrig**;  
 **this**.**historyData** = \_.cloneDeep(**this**.**OcrField**);  
 }  
 }  
};  
</**script**>

**package** com.wisdom.controller.ocr;  
  
**import** com.aspose.cells.Workbook;  
**import** com.wisdom.controller.business.CustomerApplyController;  
**import** com.wisdom.domain.ocr.dto.OcrResultDataDTO;  
**import** com.wisdom.ocr.dto.OcrOperationBackDto;  
**import** com.wisdom.ocr.dto.OcrOperationBodyDto;  
**import** com.wisdom.security.SecurityUtils;  
**import** com.wisdom.service.glusterfs.FileSystemService;  
**import** com.wisdom.service.ocr.OcrOperationService;  
**import** com.wisdom.utils.\*;  
**import** org.apache.catalina.security.SecurityUtil;  
**import** org.slf4j.LoggerFactory;  
**import** org.springframework.beans.factory.annotation.Autowired;  
**import** org.springframework.web.bind.annotation.PostMapping;  
**import** org.springframework.web.bind.annotation.RequestBody;  
**import** org.springframework.web.bind.annotation.RequestMapping;  
**import** org.springframework.web.bind.annotation.RestController;  
**import** org.springframework.web.multipart.MultipartFile;  
**import** com.aspose.words.\*;  
  
**import** java.io.\*;  
**import** java.util.Date;  
**import** java.util.Map;  
**import** java.util.List;  
**import** java.util.logging.Logger;  
  
@RestController  
@RequestMapping(**"/api/ocr"**)  
**public class** OcrController {  
 @Autowired  
 FileSystemService **fileSystemService**;  
 @Autowired  
 OcrOperationService **ocrOperationService**;  
  
 **private static** org.slf4j.Logger *logger* = LoggerFactory.*getLogger*(CustomerApplyController.**class**);  
  
  
 @PostMapping(value = **"/ocrOperation"**)  
 **public** Result ocrOperation(MultipartFile file, OcrOperationBodyDto ocrOperationBodyDto) **throws** Exception {  
 System.***out***.println(**"请求开始时间:"** + **new** Date().getTime());  
  
 String suffix = file.getOriginalFilename().substring(file.getOriginalFilename().lastIndexOf(**"."**) + 1);  
 String localTimeFileName = System.*currentTimeMillis*() + **"."** + suffix;  
 String keyName = SecurityUtils.*getCurrentCompanyId*() + **"/"**;  
 **if**( ocrOperationBodyDto.getSheetType() != **null** && !ocrOperationBodyDto.getSheetType().isEmpty() ) {  
 keyName += ocrOperationBodyDto.getSheetType() + **"/"** + localTimeFileName;  
 } **else** {  
 keyName += localTimeFileName;  
 }  
 String urlFileName = **fileSystemService**.uploadFileToBucket(file.getInputStream(), keyName, file.getSize(), **"/ocr\_rec"**);  
 System.***out***.println(**"文件上传完成:"** + **new** Date().getTime());  
 *logger*.info(**"urlFileName = "** + urlFileName);  
  
 *//文件保存到本地* FileUtils.*saveFileToServerPath*(file.getInputStream(), file.getOriginalFilename(), FileUtils.*getOcrFilePath*());  
 String fullFileName = FileUtils.*getOcrFilePath*() + **"/"** + file.getOriginalFilename();  
 OcrResultDataDTO resultDataDTO = **new** OcrResultDataDTO();  
  
  
 String htmlFileName = **""**;  
 **if** (suffix.toUpperCase().equals(**"DOC"**) || suffix.toUpperCase().equals(**"DOCX"**) || suffix.toUpperCase().equals(**"RTF"**)) {  
 *//生成并上传HTML文件* Document doc = **new** Document(fullFileName); *// word文档* HtmlSaveOptions options = **new** HtmlSaveOptions(SaveFormat.***HTML***);  
 options.setExportTextInputFormFieldAsText(**true**);  
 options.setExportImagesAsBase64(**true**);  
 htmlFileName = fullFileName.replace(**"."**+suffix, **""**) + **".html"**;  
 doc.save(htmlFileName, options);  
  
 } **else if**(suffix.toUpperCase().equals(**"XLS"**) || suffix.toUpperCase().equals(**"XLSX"**)) {  
 Workbook wb = **new** Workbook(fullFileName);*// 原始excel路径* com.aspose.cells.HtmlSaveOptions options = **new** com.aspose.cells.HtmlSaveOptions(SaveFormat.***HTML***);  
 options.setExportActiveWorksheetOnly(**true**);  
 options.setExportHiddenWorksheet(**true**);  
 options.setParseHtmlTagInCell(**true**);  
 options.setExportBogusRowData(**true**);  
 options.setPresentationPreference(**true**);  
 *// options.setExportImagesAsBase64(true);* htmlFileName = fullFileName.replace(**"."**+suffix, **""**) + **".html"**;  
 wb.save(htmlFileName, options);  
 }  
 String textarea = **""**;  
 String urlHtmlName;  
 File tmpHtmlFile;  
 FileInputStream htmlInput;  
 **if**(suffix.toUpperCase().equals(**"TXT"**)) {  
 InputStreamReader reader = **new** InputStreamReader(  
 file.getInputStream(), **"GBK"**);  
 BufferedReader br = **new** BufferedReader(reader);  
 String line ;  
 **while** ((line = br.readLine())!=**null**){  
 textarea += line + **"\n"**;  
 }  
 urlHtmlName = **fileSystemService**.uploadFileToBucket(file.getInputStream(), keyName.replace(**"."**+suffix, **""**) + **".html"**, file.getSize(), **"/ocr\_rec"**);  
 } **else** {  
 tmpHtmlFile = **new** File(htmlFileName);  
 htmlInput = **new** FileInputStream(tmpHtmlFile);  
 urlHtmlName = **fileSystemService**.uploadFileToBucket(htmlInput, keyName.replace(**"."**+suffix, **""**) + **".html"**, tmpHtmlFile.length(), **"/ocr\_rec"**);  
 *logger*.info(**"文件上传完成:"** + **new** Date().getTime());  
 tmpHtmlFile.delete();  
 }  
 *logger*.info(**"文件上传完成:"** + **new** Date().getTime());  
 *logger*.info(**"urlHtmlName: "** + urlHtmlName);  
 *//resultDataDTO.setReviewUrl(urlHtmlName);* String pageContent;  
 **if**(!suffix.toUpperCase().equals(**"TXT"**)) {  
 pageContent = HtmlUrlUtil.*getUrlHtml*(urlHtmlName, **"UTF-8"**);  
 *// 删除标志* pageContent = pageContent.replace(**"Evaluation Only. Created with Aspose.Words. Copyright 2003-2018 Aspose Pty Ltd."**, **""**);  
 pageContent = pageContent.substring(pageContent.indexOf(**"<body>"**)+6, pageContent.lastIndexOf(**"</body>"**));  
 } **else** {  
 pageContent = HtmlUrlUtil.*getUrlHtml*(urlHtmlName, **"GBK"**);  
 }  
 *//resultDataDTO.setRawHtml(pageContent);  
  
 //识别文件* **if** (suffix.toUpperCase().equals(**"DOC"**) || suffix.toUpperCase().equals(**"DOCX"**) || suffix.toUpperCase().equals(**"RTF"**)) {  
 resultDataDTO = **ocrOperationService**.wordFileRecog(fullFileName);  
 } **else if** (suffix.toUpperCase().equals(**"PDF"**)) {  
 resultDataDTO = **ocrOperationService**.ocrOperation(fullFileName);  
 } **else if** (suffix.toUpperCase().equals(**"XLS"**) || suffix.toUpperCase().equals(**"XLSX"**)) {  
 Map<String, Object> dataMap = **ocrOperationService**.excelFileRecog(fullFileName, **"1"**, **null**);  
 resultDataDTO.setData(dataMap);  
 } **else if** (suffix.toUpperCase().equals(**"TXT"**)) {  
 Map<String, Object> dataMap = **ocrOperationService**.excelFileRecog(fullFileName, **"1"**, textarea);  
 resultDataDTO.setData(dataMap);  
 }  
 **if**(resultDataDTO == **null**) {  
 resultDataDTO = **new** OcrResultDataDTO();  
 }  
 resultDataDTO.setReviewUrl(urlHtmlName);  
 resultDataDTO.setRawHtml(pageContent);  
 File docFile = **new** File(fullFileName);  
 docFile.delete();  
 **if** (suffix.toUpperCase().equals(**"TXT"**)) {  
 **if**(resultDataDTO.getData() == **null**) {  
 **return new** Result(400, **"数据中必须包含箱号!"**);  
 }  
 }  
 **return new** Result(resultDataDTO);  
 }  
  
 @PostMapping(value = **"/ocrOperationPb"**)  
 **public** Result ocrOperationPB(@RequestBody OcrOperationBodyDto ocrOperationBodyDto) **throws** Exception {  
 *//byte[] fileBytes = HexByteConverter.fromHexString(ocrOperationBodyDto.getFileHexStr());  
  
 //Long byteSize = 0L;  
  
 logger*.info(**"OcrOperationBodyDto - fileByte: {}, fileName: {}, fileSize: {}, docType: {}, orderNo: {}, sheetType: {} "**,  
 ocrOperationBodyDto.getFileHexStr(),  
 ocrOperationBodyDto.getFileName(),  
 ocrOperationBodyDto.getFileSize(),  
 ocrOperationBodyDto.getDocType(),  
 ocrOperationBodyDto.getOrderNo(),  
 ocrOperationBodyDto.getSheetType()  
 );  
  
 *logger*.info(**"请求开始时间:"** + **new** Date().getTime());  
  
 */\*InputStream in = new ByteArrayInputStream(fileBytes, 0, fileBytes.length);  
 System.out.println("fileBytes.length:" + fileBytes.length);\*/  
  
 //下载并上传原文件  
 /\*String ftpUrl = "ftp.cmlog.com";  
 Integer ftpPort = 21;  
 String ftpUser = "ZCHY";  
 String ftpPass = "ZCHY2015";  
 String ftpPathName = "DL";  
 String localPathName = FileUtils.getOcrFilePath();  
 FtpUtils ftpUtils = new FtpUtils();  
 ftpUtils.initFtpClient(ftpUrl, ftpPort, ftpUser, ftpPass );  
 ftpUtils.downloadFile(ftpPathName, ocrOperationBodyDto.getFileName(), localPathName);  
 ftpUtils.deleteFile(ftpPathName, ocrOperationBodyDto.getFileName());\*/* String localPathName = FileUtils.*getOcrFtpPath*();  
  
 String localFileName = localPathName + **"/"** + ocrOperationBodyDto.getFileName();  
 *logger*.info(**"localFileName = "** + localFileName);  
 File tmpFile = **new** File(localFileName);  
  
 FileInputStream fin = **new** FileInputStream(tmpFile);  
  
 String suffix = ocrOperationBodyDto.getFileName().substring(ocrOperationBodyDto.getFileName().lastIndexOf(**"."**) + 1);  
 String localTimeFileName = System.*currentTimeMillis*() + **"."** + suffix;  
 String keyName = SecurityUtils.*getCurrentCompanyId*() + **"/"**;  
 **if**( !ocrOperationBodyDto.getSheetType().isEmpty() ) {  
 keyName += ocrOperationBodyDto.getSheetType() + **"/"** + localTimeFileName;  
 } **else** {  
 keyName += localTimeFileName;  
 }  
  
 String urlFileName = **fileSystemService**.uploadFileToBucket(fin, keyName, tmpFile.length(), **"/ocr\_rec"**);  
 *logger*.info(**"文件上传完成:"** + **new** Date().getTime());  
 *logger*.info(**"urlFileName: "** + urlFileName);  
  
 String urlHtmlName = **""**;  
 String htmlFileName = **""**;  
 **if** (suffix.toUpperCase().equals(**"DOC"**) || suffix.toUpperCase().equals(**"DOCX"**) || suffix.toUpperCase().equals(**"RTF"**)) {  
 *//上传HTML文件* Document doc = **new** Document(localFileName); *// word文档* HtmlSaveOptions options = **new** HtmlSaveOptions(SaveFormat.***HTML***);  
 options.setExportTextInputFormFieldAsText(**true**);  
 options.setExportImagesAsBase64(**true**);  
 htmlFileName = localFileName.replaceAll(**"."** + suffix, **""**) + **".html"**;  
 doc.save(htmlFileName, options);  
  
 } **else if**(suffix.toUpperCase().equals(**"XLS"**) || suffix.toUpperCase().equals(**"XLSX"**)) {  
 Workbook wb = **new** Workbook(localFileName);*// 原始excel路径* com.aspose.cells.HtmlSaveOptions options = **new** com.aspose.cells.HtmlSaveOptions(SaveFormat.***HTML***);  
 options.setExportActiveWorksheetOnly(**true**);  
 options.setExportHiddenWorksheet(**true**);  
 options.setParseHtmlTagInCell(**true**);  
 options.setExportBogusRowData(**true**);  
 options.setPresentationPreference(**true**);  
 htmlFileName = localFileName.replace(**"."**+suffix, **""**) + **".html"**;  
 wb.save(htmlFileName, options);  
 }  
 String textarea = **""**;  
 File tmpHtmlFile;  
 FileInputStream htmlInput;  
 **if**(suffix.toUpperCase().equals(**"TXT"**)) {  
 htmlInput = **new** FileInputStream(tmpFile);  
 InputStreamReader reader = **new** InputStreamReader(htmlInput, **"GBK"**);  
 BufferedReader br = **new** BufferedReader(reader);  
 String line;  
 **while** ((line = br.readLine())!=**null**){  
 textarea += line + **"\n"**;  
 }  
 htmlFileName = localFileName.replaceAll(**"."** + suffix, **""**) + **".html"**;  
 String html = **"<html>\n"** +  
 **"<head>"** +  
 **"<meta http-equiv=\"Content-Type\" content=\"text/html; charset=gbk\" />"** +  
 **"</head>"** +  
 **"<style>"** +  
 **"\*{"** +  
 **"padding: 0;"** +  
 **"margin: 0;"** +  
 **"font-size: 12px;"** +  
 **"}"** +  
 **"</style>"** +  
 **"<body>\n"** +  
 **"<table border=\"1\">"**;  
 List<List<String>> textTable = **ocrOperationService**.*analysisTextTableData*(textarea);  
 Boolean firstRow = **true**;  
 **for** (List<String> row : textTable) {  
 html += **"<tr>"**;  
 **for**(String col: row) {  
 String text = col.replace(**"\n"**, **"</br>"**);  
 **if**(firstRow == **true**) {  
 html += **"<th>"**;  
 html += text;  
 html += **"</th>"**;  
 } **else** {  
 html += **"<td>"**;  
 html += text;  
 html += **"</td>"**;  
 }  
 }  
 **if**(firstRow == **true**) {  
 firstRow = **false**;  
 }  
 html += **"</tr>"**;  
 }  
 html += **"</table>\n"** +  
 **"</body>\n"** +  
 **"</html>"**;  
 **ocrOperationService**.*saveFile*(htmlFileName, html);  
 tmpHtmlFile = **new** File(htmlFileName);  
 htmlInput = **new** FileInputStream(tmpHtmlFile);  
 } **else** {  
 tmpHtmlFile = **new** File(htmlFileName);  
 htmlInput = **new** FileInputStream(tmpHtmlFile);  
 }  
 urlHtmlName = **fileSystemService**.uploadFileToBucket(htmlInput, keyName.replace(**"."**+suffix, **""**) + **".html"**, tmpHtmlFile.length(), **"/ocr\_rec"**);  
 *logger*.info(**"文件上传完成:"** + **new** Date().getTime());  
 *//删除本地临时文件* tmpHtmlFile.delete();  
 *//识别文件* OcrResultDataDTO resultDataDTO = **new** OcrResultDataDTO();  
 **if** (suffix.toUpperCase().equals(**"DOC"**) || suffix.toUpperCase().equals(**"DOCX"**) || suffix.toUpperCase().equals(**"RTF"**)) {  
 resultDataDTO = **ocrOperationService**.wordFileRecog(localFileName);  
 } **else if** (suffix.toUpperCase().equals(**"PDF"**)) {  
 resultDataDTO = **ocrOperationService**.ocrOperation(localFileName);  
 } **else if** (suffix.toUpperCase().equals(**"XLS"**) || suffix.toUpperCase().equals(**"XLSX"**)) {  
 Map<String, Object> dataMap = **ocrOperationService**.excelFileRecog(localFileName, **"1"**, **null**);  
 resultDataDTO.setData(dataMap);  
 } **else if** (suffix.toUpperCase().equals(**"TXT"**)) {  
 Map<String, Object> dataMap = **ocrOperationService**.excelFileRecog(localFileName, **"1"**, textarea);  
 resultDataDTO.setData(dataMap);  
 }  
 **if**(resultDataDTO == **null**) {  
 resultDataDTO = **new** OcrResultDataDTO();  
 }  
 resultDataDTO.setReviewUrl(urlHtmlName);  
 resultDataDTO.setOrderNo(ocrOperationBodyDto.getOrderNo());  
 tmpFile.delete();  
 **if** (suffix.toUpperCase().equals(**"TXT"**)) {  
 **if**(resultDataDTO.getData() == **null**) {  
 **return new** Result(400, **"数据中必须包含箱号!"**);  
 }  
 }  
 **return new** Result(resultDataDTO);  
 }  
  
 @PostMapping(value = **"/ocrOperationBack"**)  
 **public** Result ocrOperationBack(@RequestBody OcrOperationBackDto ocrOperationBackDto) **throws** Exception {  
 *logger*.info(**"回传的值：{}"**, ocrOperationBackDto);  
 **return new** Result(**true**);  
 }  
}

**package** com.wisdom.service.ocr;  
  
**import** com.aspose.cells.Cells;  
**import** com.aspose.cells.Workbook;  
**import** com.aspose.cells.Worksheet;  
**import** com.aspose.cells.WorksheetCollection;  
**import** com.github.pagehelper.util.StringUtil;  
**import** com.wisdom.domain.codec.BusinessCode;  
**import** com.wisdom.domain.ocr.OcrPreprocessingField;  
**import** com.wisdom.domain.ocr.OcrPreprocessingKeywords;  
**import** com.wisdom.domain.ocr.OcrSearchRecog;  
**import** com.wisdom.domain.ocr.dto.\*;  
**import** com.wisdom.mapper.codec.BusinessCodeMapper;  
**import** com.wisdom.ocr.dto.FieldKeywordsInfo;  
**import** com.wisdom.ocr.dto.FieldSplitDto;  
**import** com.wisdom.ocr.dto.FieldTableCellDto;  
  
**import** com.aspose.words.\*;  
  
**import** java.io.\*;  
  
  
**import** java.math.BigDecimal;  
**import** java.util.\*;  
**import** java.util.concurrent.ConcurrentHashMap;  
**import** java.util.function.Function;  
**import** java.util.function.Predicate;  
**import** java.util.regex.Matcher;  
**import** java.util.regex.Pattern;  
  
**import** com.abbyy.FREngine.\*;  
**import** com.wisdom.ocr.dto.WordTableCell;  
**import** com.wisdom.ocr.utils.CurrentGroupDto;  
**import** com.wisdom.ocr.utils.SystemConfig;  
**import** com.wisdom.service.glusterfs.FileSystemService;  
**import** net.sf.json.JSONArray;  
**import** org.apache.commons.collections.map.LinkedMap;  
**import** org.apache.commons.lang.StringUtils;  
**import** org.apache.http.client.utils.CloneUtils;  
**import** org.jsoup.nodes.Element;  
**import** org.jsoup.select.Elements;  
**import** org.slf4j.LoggerFactory;  
**import** org.springframework.beans.factory.annotation.Autowired;  
**import** org.springframework.stereotype.Service;  
**import** org.springframework.transaction.annotation.Transactional;  
  
**import** javax.annotation.PostConstruct;  
**import** java.util.List;  
**import** java.util.stream.Collectors;  
  
@Service  
@Transactional(rollbackFor = Exception.**class**)  
**public class** OcrOperationService {  
  
 */\*\*  
 \* OCR识别  
 \*  
 \** ***@param*** *\** ***@return*** *\*/* @Autowired  
 OcrFieldService **ocrFieldService**;  
 @Autowired  
 OcrFieldKeywordService **ocrFieldKeywordService**;  
 @Autowired  
 BusinessCodeMapper **businessCodeMapper**;  
 @Autowired  
 FileSystemService **fileSystemService**;  
 @Autowired  
 OcrSearchRecogService **ocrSearchRecogService**;  
 @Autowired  
 OcrPreprocessingFieldService **ocrPreprocessingFieldService**;  
 @Autowired  
 OcrPreprocessingKeywordsService **ocrPreprocessingKeywordsService**;  
 **private static** org.slf4j.Logger *logger* = LoggerFactory.*getLogger*(OcrOperationService.**class**);  
  
 *// 构建检索Map* Map<String, Map<String, String>> **matchingMap** = **new** HashMap<>();  
  
 @PostConstruct  
 **public void** initMethod() {  
 **long** keyTime = System.*currentTimeMillis*();  
 *// 增加包装类型匹配Map数据* Map<String, String> goodsKindMap = **new** HashMap<>();  
 Map<String, Object> searchMap = **new** HashMap<>();  
 searchMap.put(**"codeType"**, **"PKTP"**);  
 List<BusinessCode> goodsTypeList = **businessCodeMapper**.searchFromAll(searchMap);  
 **for** (BusinessCode item : goodsTypeList) {  
 goodsKindMap.put(item.getEnglishName(), item.getId().toString());  
 goodsKindMap.put(item.getCode(), item.getId().toString());  
 }  
 **matchingMap**.put(**"PACKAGE\_TYPE\_MAP"**, goodsKindMap);  
  
 *// 增加箱型匹配Map数据* Map<String, String> qtyMap = **new** HashMap<>();  
 Map<String, Object> qtySearchMap = **new** HashMap<>();  
 qtySearchMap.put(**"codeType"**, **"CTNCLCD"**);  
 List<BusinessCode> qtyTypeList = **businessCodeMapper**.searchFromAll(qtySearchMap);  
 **for** (BusinessCode item : qtyTypeList) {  
 qtyMap.put(item.getCode().substring(2), item.getId().toString());  
 }  
 **matchingMap**.put(**"QTY\_TYPE\_MAP"**, qtyMap);  
  
 *// 增加港口匹配Map数据* Map<String, String> portMap = **new** HashMap<>();  
 Map<String, Object> portSearchMap = **new** HashMap<>();  
 portSearchMap.put(**"codeType"**, **"PORTCD"**);  
 List<BusinessCode> portTypeList = **businessCodeMapper**.searchFromAll(portSearchMap);  
 **for** (BusinessCode item : portTypeList) {  
 portMap.put(item.getCode(), item.getCode().toString());  
 portMap.put(item.getEnglishName(), item.getCode().toString());  
 portMap.put(item.getEnglishName() + **","** + item.getCountryCode(), item.getCode().toString());  
 portMap.put(item.getEnglishName() + **","** + item.getCountryCode(), item.getCode().toString());  
 portMap.put(item.getEnglishName() + **" "** + item.getCountryCode(), item.getCode().toString());  
 portMap.put(item.getEnglishName() + **","** + item.getCountryName(), item.getCode().toString());  
 portMap.put(item.getEnglishName() + **" "** + item.getCountryName(), item.getCode().toString());  
 portMap.put(item.getEnglishName() + **","** + item.getStateCode() + **","** + item.getCountryCode(), item.getCode().toString());  
 portMap.put(item.getEnglishName() + **" "** + item.getStateCode() + **","** + item.getCountryCode(), item.getCode().toString());  
 portMap.put(item.getEnglishName() + **","** + item.getStateCode() + **","** + item.getCountryName(), item.getCode().toString());  
 portMap.put(item.getEnglishName() + **" "** + item.getStateCode() + **","** + item.getCountryName(), item.getCode().toString());  
 portMap.put(item.getEnglishName().split(**","**)[0], item.getCode().toString());  
 **if** (item.getCountryCode().equals(**"US"**)) {  
 portMap.put(item.getEnglishName() + **","** + item.getCountryCode() + **"A"**, item.getCode().toString());  
 portMap.put(item.getEnglishName() + **" "** + item.getCountryCode() + **"A"**, item.getCode().toString());  
 portMap.put(item.getEnglishName() + **","** + item.getStateCode() + **","** + item.getCountryCode() + **"A"**, item.getCode().toString());  
 portMap.put(item.getEnglishName() + **" "** + item.getStateCode() + **","** + item.getCountryCode() + **"A"**, item.getCode().toString());  
 }  
 }  
 **matchingMap**.put(**"PORT\_MAP"**, portMap);  
 System.***out***.println(**"关键字读取用时:"** + (System.*currentTimeMillis*() - keyTime + **"ms"**));  
 }  
  
 **public** OcrResultDataDTO ocrOperation(String filePath) **throws** Exception {  
 *// Load ABBYY FineReader Engine* loadEngine();  
 **try** {  
 *// Process with ABBYY FineReader Engine* **return** processWithEngine(filePath);  
 } **finally** {  
 *// Unload ABBYY FineReader Engine* unloadEngine();  
 }  
 }  
  
 **private** List<Integer> searchAllIndex(String str, String key) {  
 List<Integer> posList = **new** ArrayList<>();  
 Integer pos = str.indexOf(key);*//\*第一个出现的索引位置* **while** (pos != -1) {  
 pos = str.indexOf(key, pos + 1);*//\*从这个索引往后开始第一个出现的位置* posList.add(pos);  
 }  
 **return** posList;  
 }  
  
 */\*\*  
 \* Match the value from the cell with the keyword map, will get the value for the key word from the cell.  
 \*  
 \** ***@param fieldKeywordMap*** *\** ***@param fieldTableCellDto*** *\** ***@return*** *\*/* **private** FieldTableCellDto analysisResult(Map<Integer, FieldKeywordsInfo> fieldKeywordMap, FieldTableCellDto fieldTableCellDto) {  
 **if** (fieldTableCellDto != **null** && fieldTableCellDto.getText().trim().length() > 0) { *//不为空则执行分析* String text = fieldTableCellDto.getText();  
  
 *// Remove some strange characters from OCR recognition.* text = text.replaceAll(**"¬\n"**, **""**);  
 text = text.replaceAll(**"\\u2028|\\u2029|\\u000B|\\r"**, **"\n"**);  
 text = text.replaceAll(**"\\u0013|\\u0014|\\u0015|\\u00A0|\\uFEFF|\\u0007"**, **" "**);  
 text = text.replaceAll(**"HYPERLINK \"[\\s\\S]\*\".\*?"**, **""**);  
 fieldTableCellDto.setText(text);  
 fieldTableCellDto.setType(FieldTableCellDto.***CELL\_CONTENT\_TYPE\_CONTENT***); *// 默认为值(或者无法解析的数据)* List<FieldSplitDto> fieldSplitDtoList = **new** ArrayList<>();  
 **for** (Map.Entry<Integer, FieldKeywordsInfo> fieldEntry : fieldKeywordMap.entrySet()) {  
 FieldKeywordsInfo keywordsInfoItem = fieldEntry.getValue();  
 String searchKeywords = **""**;  
 **for** (String item : keywordsInfoItem.getKeywordsList()) {  
 String keyStr = item.replaceAll(**" "**, **"[\\\\s\\\\S]\*"**).replaceAll(**"-"**, **"\\\\s\*"**);  
 searchKeywords += **"\\b"** + keyStr + **"|"**;  
 *//searchKeywords = "[\\.]?\\b" + searchKeywords + item.replace(" ", "\\S?\\s\*[\\.]?") + "\\S?" +"|";* }  
 searchKeywords = searchKeywords.substring(0, searchKeywords.length() - 1);  
 *//String rexp = "((\\(|\\（)\*[\\s]\*([\\S]{0,2}[\\.|,|#|\*]+[ ]{0,2})?(" + searchKeywords + ")+[\\s\\S]\*?(\\)|\\）)\*(\\s\*(\\(|\\（)+[\\s\\S]\*?(\\)|\\）)+)\*\\s\*(\\.|,|#|&|and|or|/)\*\\s\*(:|：|\\r|\\n|\\s{3,})\*)+";  
 //String rexp = (isHead?"(^|\\n)":"") + "((\\(|\\（)\*[\\s]\*([\\S]{0,2}[\\.|,|#|\*]+[ ]?)?(" + searchKeywords + ")+[\\s\\S]\*?(\\)|\\）)\*(\\s\*(\\(|\\（)+[\\s\\S]\*?(\\)|\\）)+)\*\\s\*(\\.|,|#|&|and|or|/|；|;)\*)+(:|：|\\r|\\n| {3,})+";  
 //最终0927 String rexp = (isHead?"(^|\\n)":"") + "((\\(|\\（)\*\\s\*(\\S{0,2}[\\.,#\*]+( )?)?(" + searchKeywords + ")+[\\s\\S]\*?(\\)|\\）)\*(\\s\*(\\(|\\（)+[\\s\\S]\*?(\\)|\\）)+)\*\\s\*(\\.|,|#|&|and|or|/|;)\*\\s\*(:|：|\\r|\\n|\\s{3,})\*)+(:|：|\\r|\\n|\\s\*|$)";  
 // 修改为更模糊的匹配方式，取消了只有括号的限制，匹配到换行符 冒号 制表符即为结束标志* String rexp;  
 **if** (keywordsInfoItem.getMatchingMode() == keywordsInfoItem.*MATCHING\_MODE\_TAIL\_STRICT*) {  
 rexp = (keywordsInfoItem.getHeaderStartFlag() ? **"(^|\\n)"** : **""**) + **"((\\(|\\（)\*\\s\*(\\S{0,2}[\\.,#\*]+( )?)?("** + searchKeywords + **")+\\s\*(\\)|\\）)\*(\\s\*[(（]+[\\s\\S]\*?[)|）]+)\*[:：\\r\\n\\s]\*)+(:|：|\\t|\\r|\\n|$)+"**;  
 } **else** {  
 rexp = (keywordsInfoItem.getHeaderStartFlag() ? **"(^|\\n)"** : **""**) + **"((\\(|\\（)\*\\s\*(\\S{0,2}[\\.,#\*]+( )?)?("** + searchKeywords + **")+[\\s\\S]\*?(\\)|\\）)\*(\\s\*[(（]+[\\s\\S]\*?[)|）]+)\*[:：\\r\\n\\s]\*)+(:|：|\\t|\\r|\\n|$)+"**;  
 }  
 *//String rexp = (isHead?"(^|\\n)":"") + "((\\(|\\（)\*\\s\*(\\S{0,2}[\\.,#\*]+( )?)?(" + searchKeywords + ")+[\\s\\S]\*?(\\)|\\）)\*(\\s\*(\\(|\\（)+[\\s\\S]\*?(\\)|\\）)+)\*\\s\*(\\.|,|#|&|and|or|/|;)\*\\s\*(:|：|\\r|\\n|\\s{3,})\*)+(:|：|\\\\t|\\\\r|\\\\n|$)+";* Pattern pattern = Pattern.*compile*(rexp, Pattern.***CASE\_INSENSITIVE***);  
  
 Matcher matcher = pattern.matcher(text);  
 **while** (matcher.find()) {  
 String fieldStr = matcher.group();  
 *// 对匹配到的字段名再次进行正则匹配，区分是否为多字段同行。* Integer index = text.indexOf(fieldStr);  
 **if** (index > 0) {  
 *//* ***TODO to be tested, if need ……* if** (Pattern.*matches*(**"^\\s\*\\n+[\\s\\S]\*"**, fieldStr)) { *// 包含换行开头则为上下结构(纵向拆分)* fieldSplitDtoList.add(**new** FieldSplitDto(fieldEntry.getKey(), index, fieldStr, **"Y"**));  
 } **else** { *// 否则为横向换行(水平拆分)* fieldSplitDtoList.add(**new** FieldSplitDto(fieldEntry.getKey(), index, fieldStr, **"X"**));  
 }  
 } **else** { *// 开头不需要拆分* fieldTableCellDto.setKeyStr(fieldStr);  
 *//尝试返回结果 后期废弃* String[] result = *splitByStr*(text, fieldStr);  
 **if** (result.**length** > 1) {  
 fieldTableCellDto.setValueStr(result[1].trim());  
 fieldTableCellDto.setType(**"KV"**); *// 单元格包含标题和值* fieldTableCellDto.setFinish(**true**);  
 } **else** {  
 fieldTableCellDto.setType(**"K"**); *// 单元格只包含标题* fieldTableCellDto.setValidKey(fieldEntry.getValue().getValidKey());  
 }  
 fieldTableCellDto.setFieldId(fieldEntry.getKey()); *//设置字段id* }  
 }  
  
 *//* ***TODO 拆分数据* if** (fieldSplitDtoList.size() > 0) {  
 fieldTableCellDto.setFieldSplitList(fieldSplitDtoList);  
 }  
 **if** (fieldTableCellDto.getType().equals(**"V"**)) {  
 fieldTableCellDto.setValueStr(fieldTableCellDto.getText().trim());  
 }  
  
 }  
 **return** fieldTableCellDto;  
 }  
 **return null**;  
 }  
  
  
 **private void** loadEngine() **throws** Exception {  
 *displayMessage*(**"Initializing Engine..."**);  
 **engine** = Engine.*InitializeEngine*(SystemConfig.*GetDllFolder*(), SystemConfig.*GetCustomerProjectId*(),  
 SystemConfig.*GetLicensePath*(), SystemConfig.*GetLicensePassword*(), **""**, **""**, **false**);  
  
*// engine.LoadPredefinedProfile("DocumentConversion\_Accuracy");  
  
 // Setup the language.  
// setTextLanguages();  
  
 // When print the language  
// printPredefinedTextLanguages();* }  
  
  
 **private** OcrResultDataDTO processWithEngine(String filePath) {  
 **try** {  
 *// Setup FREngine* setupFREngine();  
  
 *// Process sample image* **return** processImage(filePath);  
 } **catch** (Exception ex) {  
 *displayMessage*(ex.getMessage());  
 }  
 **return null**;  
 }  
  
 **private void** setupFREngine() {  
 *displayMessage*(**"Loading predefined profile..."**);  
 **engine**.LoadPredefinedProfile(**"DocumentConversion\_Accuracy"**);  
 *// Possible profile names are:  
 // "DocumentConversion\_Accuracy", "DocumentConversion\_Speed",  
 // "DocumentArchiving\_Accuracy", "DocumentArchiving\_Speed",  
 // "BookArchiving\_Accuracy", "BookArchiving\_Speed",  
 // "TextExtraction\_Accuracy", "TextExtraction\_Speed",  
 // "FieldLevelRecognition",  
 // "BarcodeRecognition\_Accuracy", "BarcodeRecognition\_Speed",  
 // "HighCompressedImageOnlyPdf",  
 // "BusinessCardsProcessing",  
 // "EngineeringDrawingsProcessing",  
 // "Version9Compatibility",  
 // "Default"* }  
  
  
 **public** OcrResultDataDTO iteratePage(IFRPage page) {  
 *// 得到需要匹配的字段信息* OcrFieldDTO ocrFieldDTO = **new** OcrFieldDTO();  
 ocrFieldDTO.setSheetType(**"0"**);  
 List<OcrFieldDTO> ocrFieldDTOList = **ocrFieldService**.selectSelective(ocrFieldDTO);  
 Map<Integer, OcrFieldDTO> fieldMap = **new** HashMap<>();  
 **for** (OcrFieldDTO item : ocrFieldDTOList) {  
 fieldMap.put(item.getId(), item);  
 }  
 *// 得到关键字* List<OcrFieldKeywordDTO> fieldKeywordList = **ocrFieldKeywordService**.selectBySheetType(**"0"**);  
 Map<Integer, List<String>> fieldKeywordMap = **new** HashMap<>();  
  
  
 ILayout layout = page.getLayout();  
  
 *//* IRegion layoutRegion = layout.getPageRegion();  
  
 *// Separator block.  
  
 // Handle blocks.* ILayoutBlocks layoutBlocks = layout.getBlocks();  
 List<Integer> xList = **new** ArrayList<>();  
 List<Integer> yList = **new** ArrayList<>();  
 List<FieldTableCellDto> fieldTableCellDtoList = **new** ArrayList<>();  
 **for** (**int** i = 0; i < layoutBlocks.getCount(); ++i) {  
 IBlock block = layoutBlocks.Item(i);  
 BlockTypeEnum type = block.getType();  
 **if** (type.getValue() == ***BT\_TEXT***) {  
 ITextBlock textBlock = block.GetAsTextBlock();  
 IParagraphs paragraphs = textBlock.getText().getParagraphs();  
 String text = **""**;  
 **int** paraCount = paragraphs.getCount();  
 FieldTableCellDto fieldTableCellDto = **new** FieldTableCellDto();  
 **for** (**int** iParagraph = 0; iParagraph < paraCount; iParagraph++) {  
 IParagraph paragraph = paragraphs.Item(iParagraph);  
 text += paragraph.getText();  
 **if** (fieldTableCellDto.getLeft() != **null**) {  
 **if** (fieldTableCellDto.getLeft() > paragraph.getLeft()) {  
 fieldTableCellDto.setLeft(paragraph.getLeft());  
 }  
 } **else** {  
 fieldTableCellDto.setLeft(paragraph.getLeft());  
 }  
 **if** (fieldTableCellDto.getRight() != **null**) {  
 **if** (fieldTableCellDto.getRight() < paragraph.getRight()) {  
 fieldTableCellDto.setRight(paragraph.getRight());  
 }  
 } **else** {  
 fieldTableCellDto.setRight(paragraph.getRight());  
 }  
 **if** (fieldTableCellDto.getTop() != **null**) {  
 **if** (fieldTableCellDto.getTop() > paragraph.getTop()) {  
 fieldTableCellDto.setTop(paragraph.getTop());  
 }  
 } **else** {  
 fieldTableCellDto.setTop(paragraph.getTop());  
 }  
 **if** (fieldTableCellDto.getBottom() != **null**) {  
 **if** (fieldTableCellDto.getBottom() < paragraph.getBottom()) {  
 fieldTableCellDto.setBottom(paragraph.getBottom());  
 }  
 } **else** {  
 fieldTableCellDto.setBottom(paragraph.getBottom());  
 }  
 }  
 xList.add(fieldTableCellDto.getLeft());  
 yList.add(fieldTableCellDto.getTop());  
 fieldTableCellDto.setText(text);  
 fieldTableCellDtoList.add(fieldTableCellDto);  
 } **else if** (type.getValue() == ***BT\_TABLE***) {  
 ITableBlock tableBlock = block.GetAsTableBlock();  
 ITableCells cells = tableBlock.getCells();  
 **for** (**int** j = 0; j < cells.getCount(); ++j) {  
 ITableCell cell = cells.Item(j);  
 IBlock cellBlock = cell.getBlock();  
 ITextBlock textBlock = cellBlock.GetAsTextBlock();  
 IParagraphs paragraphs = textBlock.getText().getParagraphs();  
 String text = **""**;  
 **for** (**int** iParagraph = 0; iParagraph < paragraphs.getCount(); iParagraph++) {  
 IParagraph paragraph = paragraphs.Item(iParagraph);  
 text += paragraph.getText();  
 }  
 FieldTableCellDto fieldTableCellDto = **new** FieldTableCellDto(cell.getTop(), cell.getBottom(), cell.getLeft(), cell.getRight(), text);  
 fieldTableCellDtoList.add(fieldTableCellDto);  
  
 }  
 }  
 }  
  
 **return** recognitionCore(fieldTableCellDtoList, yList, xList);  
 }  
  
 */\*\*  
 \* 构建word 数据结构表  
 \*  
 \** ***@param filePath*** *\** ***@return*** *\*/* **public** OcrResultDataDTO wordFileRecog(String filePath) **throws** Exception {  
  
 **if** (!*getLicense*()) { *// 验证License 若不验证则转化出的pdf文档有水印* **throw new** Exception(**"com.aspose.words lic ERROR!"**);  
 }  
 **try** {  
 **long** old = System.*currentTimeMillis*();  
 Document doc = **new** Document(filePath); *// word文档  
 // 文本框读取* NodeCollection shapes = doc.getChildNodes(NodeType.***SHAPE***, **true**);  
 *// 图文框读取* **int** frameFormatCount = 0;  
 ParagraphCollection paragraphs = doc.getFirstSection().getBody().getParagraphs();  
 String text = **""**; *// 除表格外内容* **for** (Paragraph paragraph : paragraphs) {  
 **if** (paragraph.getFrameFormat().isFrame()) {  
 frameFormatCount++;  
 }  
 text += paragraph.getText() + **"\n"**;  
 }  
 text = text.replaceAll(**"\\f"**, **"\n"**);  
 *//如果文本框或图文框 > 5个 则执行转PDF识别* **if** (shapes.getCount() > 5 || frameFormatCount > 5) {  
 *//* ***TODO 转到pdf识别* return null**;  
 }  
  
 System.***out***.println(**"图文框数量:"** + frameFormatCount);  
 *// 表格读取* NodeCollection tables = doc.getChildNodes(NodeType.***TABLE***, **true**);  
 **int** rowTotal = 0;  
 **int** colTotal = 0;  
 List<WordTableCell> wordTableCellList = **new** ArrayList<>();  
 */\* for (Table table : (Iterable<Table>) tables) {  
 int tableIndex = table.getParentNode().getChildNodes().indexOf(table);  
 int rowCount = table.getRows().getCount();  
 for (Row row : table.getRows()) {  
 int rowIndex = table.getRows().indexOf(row);  
 for (Cell cell : row.getCells()) {  
 int cellIndex = row.getCells().indexOf(cell);  
 int colCount = row.getCount();  
 if (colTotal < colCount) {  
 colTotal = colCount;  
 }  
 System.out.println(cell.getCellFormat().getWidth() + "X-merge:" + cell.getCellFormat().getHorizontalMerge() + "----Y-merge:" + cell.getCellFormat().getVerticalMerge() + "----Row-Height:" + row.getRowFormat().getHeight() + "----Cell-Width:" + cell.getCellFormat().getWidth());  
 System.out.println("table:" + tableIndex + "------row:" + rowIndex + "------col:" + cellIndex + "-----rows:" + rowCount + "--------cells:" + colCount+"----context:" + cell.getText());  
 wordTableCellList.add(new WordTableCell(tableIndex > 0 ? (rowTotal + rowIndex + 1) : (rowTotal + rowIndex), cellIndex, cell.getText().trim()));  
 }  
 }  
 rowTotal += rowCount + 1; // 两个表格间空一行  
 }\*/* List<Integer> yList = **new** ArrayList<>();  
 List<Integer> xList = **new** ArrayList<>();  
 List<FieldTableCellDto> fieldTableCellDtoList = **new** ArrayList<>();  
 List<Integer> rowHeightArray = **new** ArrayList<>(); *//行高记录* **for** (Table table : (Iterable<Table>) tables) {  
 **int** tableIndex = table.getParentNode().getChildNodes().indexOf(table);  
 **int** rowCount = table.getRows().getCount();  
 **for** (Row row : table.getRows()) {  
 **int** rowIndex = table.getRows().indexOf(row);  
 Integer rowWidth = 0; *// 行宽度记录* **for** (Cell cell : row.getCells()) {  
 **int** cellIndex = row.getCells().indexOf(cell);  
 **int** colCount = row.getCount();  
 **if** (colTotal < colCount) {  
 colTotal = colCount;  
 }  
 FieldTableCellDto cellDto = **new** FieldTableCellDto();  
 **if** (rowHeightArray.size() <= cellIndex) {  
 rowHeightArray.add(0);  
 }  
 cellDto.setLeft(rowWidth);  
 rowWidth = rowWidth + (**int**) Math.*round*(cell.getCellFormat().getWidth()); *// 增加宽度* cellDto.setRight(rowWidth);  
  
 *// rowTotal + rowIndex to handle multi-table case.* cellDto.setTop((rowTotal + rowIndex) \* 100);  
 cellDto.setBottom((rowTotal + rowIndex) \* 100 + 100);  
 cellDto.setWidth(1);  
 cellDto.setHeight(1);  
 cellDto.setText(cell.getText());  
 fieldTableCellDtoList.add(cellDto);  
 yList.add(cellDto.getTop());  
 xList.add(cellDto.getLeft());  
 }  
 }  
 rowTotal += rowCount + 1; *// 两个表格间空一行* }  
  
 *//rowTotal = rowTotal - 1; // 减掉最后一个表格底部空行  
 // 添加非表格数据作为一个单元格* **if** (StringUtil.*isNotEmpty*(text)) {  
 FieldTableCellDto cellDto = **new** FieldTableCellDto();  
 **if** (fieldTableCellDtoList.size() > 0) {  
 cellDto.setText(text);  
 cellDto.setTop(yList.get(yList.size() - 1) + 1000);  
 cellDto.setBottom(yList.get(yList.size() - 1) + 2000);  
 cellDto.setLeft(xList.get(xList.size() - 1) + 1000);  
 cellDto.setRight(xList.get(xList.size() - 1) + 2000);  
 cellDto.setHeight(1);  
 cellDto.setWidth(1);  
 yList.add(yList.get(yList.size() - 1) + 1000);  
 } **else** {  
 cellDto.setText(text);  
 cellDto.setTop(0);  
 cellDto.setBottom(1000);  
 cellDto.setLeft(0);  
 cellDto.setRight(1000);  
 cellDto.setHeight(1);  
 cellDto.setWidth(1);  
 yList.add(0);  
 }  
  
 fieldTableCellDtoList.add(cellDto);  
 }  
  
 **return** recognitionCore(fieldTableCellDtoList, yList, xList);  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 }  
 **return null**;  
 }  
  
 */\*\*  
 \* 核心识别方法  
 \*  
 \** ***@param fieldTable*** *\** ***@param rowCount*** *\** ***@param colCount*** *\** ***@return*** *\*/* **private** OcrResultDataDTO recognitionCore(List<FieldTableCellDto> fieldTableCellDtoList, List<Integer> yList, List<Integer> xList) {  
 System.***out***.println(**"开始处理结果:"** + **new** Date().getTime());  
 *// 得到需要匹配的字段信息* OcrFieldDTO ocrFieldDTO = **new** OcrFieldDTO();  
 ocrFieldDTO.setSheetType(**"0"**);  
 List<OcrFieldDTO> ocrFieldDTOList = **ocrFieldService**.selectSelective(ocrFieldDTO);  
 Map<Integer, OcrFieldDTO> fieldMap = **new** HashMap<>();  
 **for** (OcrFieldDTO item : ocrFieldDTOList) {  
 fieldMap.put(item.getId(), item);  
 }  
 *// 得到关键字* List<OcrFieldKeywordDTO> fieldKeywordList = **ocrFieldKeywordService**.selectBySheetType(**"0"**);  
 Map<Integer, FieldKeywordsInfo> fieldKeywordMap = **new** HashMap<>();  
  
 **for** (OcrFieldKeywordDTO item : fieldKeywordList) {  
 **if** (fieldKeywordMap.get(item.getFieldId()) != **null**) {  
 fieldKeywordMap.get(item.getFieldId()).getKeywordsList().add(item.getKwValue());  
 } **else** {  
 FieldKeywordsInfo fieldKeywordsInfo = **new** FieldKeywordsInfo();  
 List<String> stringList = **new** ArrayList<>();  
 stringList.add(item.getKwValue());  
 fieldKeywordsInfo.setKeywordsList(stringList);  
 fieldKeywordsInfo.setHeaderStartFlag(item.getHeaderFlag());  
 fieldKeywordMap.put(item.getFieldId(), fieldKeywordsInfo);  
 }  
 }  
  
 *// 对关键字按长度排序，长的先匹配 短的后匹配* **for** (**int** index = 0; index < fieldTableCellDtoList.size(); index++) {  
 FieldTableCellDto fieldTableCellDto = fieldTableCellDtoList.get(index);  
 FieldTableCellDto analysisResultCell = analysisResult(fieldKeywordMap, fieldTableCellDto);  
 **if** (analysisResultCell != **null** && analysisResultCell.getFieldSplitList() != **null** && analysisResultCell.getFieldSplitList().size() > 0) {  
 fieldTableCellDtoList.remove(index);  
 List<FieldTableCellDto> cellSplitResultList = cellDataSplit(analysisResultCell);  
 fieldTableCellDtoList.addAll(index, cellSplitResultList);  
 index += cellSplitResultList.size() - 1;  
 }  
 }  
 **for** (FieldTableCellDto fieldTableCellDto : fieldTableCellDtoList) {  
 xList.add(fieldTableCellDto.getLeft());  
 yList.add(fieldTableCellDto.getTop());  
 }  
 Set xSet = **new** HashSet(xList);  
 Set ySet = **new** HashSet(yList);  
 xList = **new** ArrayList(xSet);  
 yList = **new** ArrayList(ySet);  
 Collections.*sort*(xList);  
 Collections.*sort*(yList);  
 List<Integer> xAvgList = **new** ArrayList<>();  
 List<Integer> yAvgList = **new** ArrayList<>();  
 xAvgList.add(xList.get(0));  
 yAvgList.add(yList.get(0));  
 **for** (**int** i = 0; i < xList.size(); i++) {  
 Integer x = xList.get(i);  
 Boolean mergeFlag = **false**;  
 **for** (**int** j = 0; j < xAvgList.size(); j++) {  
 Integer xAvg = xAvgList.get(j);  
 **if** (Math.*abs*(xAvg - x) <= 15) {  
 xAvgList.set(j, (x + xAvg) / 2);  
 mergeFlag = **true**;  
 **break**;  
 }  
 }  
 **if** (!mergeFlag) {  
 xAvgList.add(x);  
 }  
 }  
 **for** (**int** i = 0; i < yList.size(); i++) {  
 Integer y = yList.get(i);  
 Boolean mergeFlag = **false**;  
 **for** (**int** j = 0; j < yAvgList.size(); j++) {  
 Integer yAvg = yAvgList.get(j);  
 **if** (Math.*abs*(yAvg - y) <= 15) {  
 yAvgList.set(j, (y + yAvg) / 2);  
 mergeFlag = **true**;  
 **break**;  
 }  
 }  
 **if** (!mergeFlag) {  
 yAvgList.add(y);  
 }  
 }  
 FieldTableCellDto[][] fieldTable = **new** FieldTableCellDto[yAvgList.size()][xAvgList.size()];  
 **for** (FieldTableCellDto fieldTableCellDto : fieldTableCellDtoList) {  
 **if** (StringUtil.*isNotEmpty*(fieldTableCellDto.getText().trim())) {  
 Integer xIndex = *getApproximateIndex*(xAvgList, fieldTableCellDto.getLeft());  
 Integer yIndex = *getApproximateIndex*(yAvgList, fieldTableCellDto.getTop());  
 fieldTableCellDto.setWidth(**this**.*getBoxAxisDis*(xAvgList, fieldTableCellDto.getRight()) - xIndex + 1);  
 fieldTableCellDto.setHeight(**this**.*getBoxAxisDis*(yAvgList, fieldTableCellDto.getBottom()) - yIndex + 1);  
 fieldTable[yIndex][xIndex] = fieldTableCellDto;  
 }  
 }  
  
 *// 构建指针* **for** (Integer y = 0; y < yAvgList.size(); y++) {  
 **for** (Integer x = 0; x < xAvgList.size(); x++) {  
 **if** (fieldTable[y][x] != **null**) {  
 FieldTableCellDto item = fieldTable[y][x];  
 **for** (Integer row = y; row < item.getHeight(); row++) {  
 **for** (Integer col = x; col < item.getWidth(); col++) {  
 **if** (row != y || col != x) {  
  
 }  
 }  
 }  
 }  
 }  
 }  
  
 *// 过滤纵向空数据 直接向下匹配到数据则为合并完成* **for** (Integer y = 0; y < yAvgList.size(); y++) {  
 **for** (Integer x = 0; x < xAvgList.size(); x++) {  
 **if** (fieldTable[y][x] != **null**) {  
 FieldTableCellDto item = fieldTable[y][x];  
 **if** (fieldTable[y][x].getType().equals(**"K"**) || fieldTable[y][x].getType().equals(**"V"**)) {  
 **for** (Integer row = (y + item.getHeight()); row < yAvgList.size(); row++) {  
 **if** (fieldTable[row][x] == **null**) { *// 下方如果为空* item.setHeight(item.getHeight() + 1);  
 item.setBottom(yAvgList.get(row));  
 } **else** {  
 **break**;  
 }  
 }  
 }  
 }  
 }  
 }  
  
 **long** dataStartTime = System.*currentTimeMillis*();  
 *// 对数据进行匹配 获取业务code （前提：数据分割比较完善）* **for** (Integer y = 0; y < yAvgList.size(); y++) {  
 **for** (Integer x = 0; x < xAvgList.size(); x++) {  
 **if** (fieldTable[y][x] != **null**) {  
 *// 对未知结果进行匹配* **if** (fieldTable[y][x].getType().equals(**"V"**) && fieldTable[y][x].getBusinessType() == **null**) {  
 **for** (OcrFieldDTO fieldDTO : ocrFieldDTOList) {  
 *// 不能直接匹配 但可以进行匹配（对值进行匹配，例如：港口）* **if** (fieldDTO.getDirectSearchFlag() == **false** && fieldDTO.getRecogType() != **null** && fieldDTO.getRecogType() == ***REG\_PRE\_MAP***) {  
 **if** (StringUtil.*isNotEmpty*(fieldDTO.getRecogStrategy())) {  
 String[] splitStrings = *splitByStr*(fieldDTO.getPreStrategy(), **"####"**);  
 String valueStr = fieldTable[y][x].getValueStr().trim().replaceAll(**"\\s+"**, **" "**);  
 valueStr = valueStr.replaceAll(splitStrings[0], splitStrings[1]);  
 **long** portStartTime = System.*currentTimeMillis*();  
 String resultCode = **matchingMap**.get(fieldDTO.getRecogStrategy()).get(valueStr.toUpperCase());  
 System.***out***.println(**"港口比对用时:"** + (System.*currentTimeMillis*() - portStartTime) + **"ms"**);  
 **if** (StringUtil.*isNotEmpty*(resultCode)) { *// 匹配成功* fieldTable[y][x].setBusinessType(fieldDTO.getRecogStrategy()); *// 将匹配到的业务类型放入"V"Dto中* fieldTable[y][x].setBusinessCode(resultCode); *// 将匹配到的业务代码放入"V"Dto中* **break**;  
 }  
 }  
 }  
 }  
 } **else if** (fieldTable[y][x].getType().equals(**"KV"**)) { *//对已经分割结果进行匹配* OcrFieldDTO fieldDTO = fieldMap.get(fieldTable[y][x].getFieldId());  
 **if** (fieldDTO.getRecogType() != **null** && fieldDTO.getRecogType() == ***REG\_PRE\_MAP***) { *// 如果需要匹配* String[] splitStrings = *splitByStr*(fieldDTO.getPreStrategy(), **"####"**);  
 String valueStr = fieldTable[y][x].getValueStr().trim().replaceAll(**"\\s+"**, **" "**);  
 valueStr = valueStr.replaceAll(splitStrings[0], splitStrings[1]);  
 **if** (StringUtil.*isNotEmpty*(fieldDTO.getRecogStrategy())) {  
 **long** portStartTime = System.*currentTimeMillis*();  
 String resultCode = **matchingMap**.get(fieldDTO.getRecogStrategy()).get(valueStr.toUpperCase());  
 System.***out***.println(**"港口比对用时:"** + (System.*currentTimeMillis*() - portStartTime) + **"ms"**);  
 **if** (StringUtil.*isNotEmpty*(resultCode)) { *// 匹配成功* fieldTable[y][x].setBusinessType(fieldDTO.getRecogStrategy()); *// 将匹配到的业务类型放入"KV"Dto中* fieldTable[y][x].setBusinessCode(resultCode); *// 将匹配到的业务代码放入"KV"Dto中* } **else** {  
 fieldTable[y][x].setValueStr(**""**); *//匹配不成功 则清除结果* }  
 }  
 }  
 } **else if** (fieldTable[y][x].getType().equals(**"K"**)) { *// 对标题设置业务类型* OcrFieldDTO fieldDTO = fieldMap.get(fieldTable[y][x].getFieldId());  
 **if** (fieldDTO.getDirectSearchFlag() == **false** && fieldDTO.getRecogType() != **null** && fieldDTO.getRecogType() == ***REG\_PRE\_MAP***) {  
 fieldTable[y][x].setBusinessType(fieldMap.get(fieldTable[y][x].getFieldId()).getRecogStrategy());  
 }  
  
 }  
 }  
 }  
 }  
  
 **for** (Integer y = 0; y < yAvgList.size(); y++) {  
 **for** (Integer x = 0; x < xAvgList.size(); x++) {  
 **if** (fieldTable[y][x] != **null**) {  
 *// 可优化为多次遍历匹配  
 // 底部格子比对结果* Boolean underBoxFlag = **false**;  
 *// 底部类型比对结果* Boolean underTypeFlag = **false**;  
 *// 右侧格子比对结果* Boolean nextBoxFlag = **false**;  
 *// 右侧类型比对结果* Boolean nextTypeFlag = **false**;  
 *// cell为需要业务类型匹配格子* Boolean isTypeFlag = **false**;  
 **if** (fieldTable[y][x].getFinish() == **false** && fieldTable[y][x].getType().equals(**"K"**)) {  
 **if** (y + fieldTable[y][x].getHeight() < yAvgList.size()) {  
 *// 标题格子下方的格子必须是没有完成 并且是"值";(可优化查询)* **if** (fieldTable[y + fieldTable[y][x].getHeight()][x] != **null** && fieldTable[y + fieldTable[y][x].getHeight()][x].getFinish() == **false** && fieldTable[y + fieldTable[y][x].getHeight()][x].getType().equals(**"V"**)) {  
 **if** (StringUtil.*isNotEmpty*(fieldTable[y][x].getBusinessType())) {  
 **if** (StringUtil.*isNotEmpty*(fieldTable[y + fieldTable[y][x].getHeight()][x].getBusinessType()) && fieldTable[y][x].getBusinessType().equals(fieldTable[y + fieldTable[y][x].getHeight()][x].getBusinessType())) {  
 underTypeFlag = **true**;  
 } **else** {  
 underTypeFlag = **false**;  
 }  
 isTypeFlag = **true**;  
 } **else** {  
 underTypeFlag = **false**;  
 }  
 *// 判断格子是否对齐相等  
 /\*if (fieldTable[y][x].getWidth() == fieldTable[y + fieldTable[y][x].getHeight()][x].getWidth()) {\*/* underBoxFlag = **true**;  
 */\*} else {  
 underBoxFlag = false;  
 }\*/* }  
 }  
 **if** (x + fieldTable[y][x].getWidth() < xAvgList.size()) {  
 **if** (fieldTable[y][x + fieldTable[y][x].getWidth()] != **null** && fieldTable[y][x + fieldTable[y][x].getWidth()].getFinish() == **false** && fieldTable[y][x + fieldTable[y][x].getWidth()].getType().equals(**"V"**)) {  
 **if** (StringUtil.*isNotEmpty*(fieldTable[y][x].getBusinessType())) {  
 **if** (StringUtil.*isNotEmpty*(fieldTable[y][x + fieldTable[y][x].getWidth()].getBusinessType()) && fieldTable[y][x].getBusinessType().equals(fieldTable[y][x + fieldTable[y][x].getWidth()].getBusinessType())) {  
 nextTypeFlag = **true**;  
 } **else** {  
 nextTypeFlag = **false**;  
 }  
 isTypeFlag = **true**;  
 } **else** {  
 nextTypeFlag = **false**;  
 }  
 *// 判断格子是否对齐相等  
 /\*if (fieldTable[y][x].getHeight() == fieldTable[y][x + fieldTable[y][x].getWidth()].getHeight()) {\*/* nextBoxFlag = **true**;  
 */\*} else {  
 nextBoxFlag = false;  
 }\*/* }  
 }  
 }  
 *//  
 // 如果下格类型匹配 右侧不匹配 则直接使用下格作为数据  
 // 如果下格类型不匹配 右侧匹配 则直接使用右格作为数据  
 // 都为空则认为不是结果 不匹配直接跳出* **if** (isTypeFlag) {  
 **if** (underTypeFlag && !nextTypeFlag) {  
 fieldTable[y][x].setValueStr(fieldTable[y + fieldTable[y][x].getHeight()][x].getValueStr()); *// 取到新值赋值给标题* fieldTable[y][x].setBottom(fieldTable[y + fieldTable[y][x].getHeight()][x].getBottom()); *// 取得值区域底部位置* fieldTable[y][x].setRight(fieldTable[y + fieldTable[y][x].getHeight()][x].getRight()); *// 取得值区域右侧位置* fieldTable[y][x].setType(**"KV"**);  
 fieldTable[y][x].setFinish(**true**);  
 fieldTable[y + fieldTable[y][x].getHeight()][x].setFinish(**true**);  
 } **else if** (!underTypeFlag && nextTypeFlag) {  
 fieldTable[y][x].setValueStr(fieldTable[y][x + fieldTable[y][x].getWidth()].getValueStr());  
 fieldTable[y][x].setBottom(fieldTable[y][x + fieldTable[y][x].getWidth()].getBottom()); *// 取得值区域底部位置* fieldTable[y][x].setRight(fieldTable[y][x + fieldTable[y][x].getWidth()].getRight()); *// 取得值区域右侧位置* fieldTable[y][x].setType(**"KV"**);  
 fieldTable[y][x].setFinish(**true**);  
 fieldTable[y][x + fieldTable[y][x].getWidth()].setFinish(**true**);  
 }  
 **continue**;  
 }  
  
  
 **if** (underBoxFlag && !nextBoxFlag) {  
 fieldTable[y][x].setValueStr(fieldTable[y + fieldTable[y][x].getHeight()][x].getValueStr()); *// 取到新值赋值给标题* fieldTable[y][x].setBottom(fieldTable[y + fieldTable[y][x].getHeight()][x].getBottom()); *// 取得值区域底部位置* fieldTable[y][x].setRight(fieldTable[y + fieldTable[y][x].getHeight()][x].getRight()); *// 取得值区域右侧位置* fieldTable[y][x].setType(**"KV"**);  
 fieldTable[y][x].setFinish(**true**);  
 fieldTable[y + fieldTable[y][x].getHeight()][x].setFinish(**true**);  
 } **else if** (!underBoxFlag && nextBoxFlag) {  
 fieldTable[y][x].setValueStr(fieldTable[y][x + fieldTable[y][x].getWidth()].getValueStr());  
 fieldTable[y][x].setBottom(fieldTable[y][x + fieldTable[y][x].getWidth()].getBottom()); *// 取得值区域底部位置* fieldTable[y][x].setRight(fieldTable[y][x + fieldTable[y][x].getWidth()].getRight()); *// 取得值区域右侧位置* fieldTable[y][x].setType(**"KV"**);  
 fieldTable[y][x].setFinish(**true**);  
 fieldTable[y][x + fieldTable[y][x].getWidth()].setFinish(**true**);  
 } **else if** (underBoxFlag) {  
 fieldTable[y][x].setValueStr(fieldTable[y + fieldTable[y][x].getHeight()][x].getValueStr()); *// 取到新值赋值给标题* fieldTable[y][x].setBottom(fieldTable[y + fieldTable[y][x].getHeight()][x].getBottom()); *// 取得值区域底部位置* fieldTable[y][x].setRight(fieldTable[y + fieldTable[y][x].getHeight()][x].getRight()); *// 取得值区域右侧位置* fieldTable[y][x].setType(**"KV"**);  
 fieldTable[y][x].setFinish(**true**);  
 fieldTable[y + fieldTable[y][x].getHeight()][x].setFinish(**true**);  
 }  
 }  
 }  
 }  
 System.***out***.println(**"数据比对用时:"** + (System.*currentTimeMillis*() - dataStartTime) + **"ms"**);  
  
 OcrResultDataDTO ocrResultDataDTO = **new** OcrResultDataDTO();  
 Map<String, FieldTableCellDto> dataMap = **new** HashMap<>();  
 String pageText = **""**;  
 **for** (Integer y = 0; y < yAvgList.size(); y++) {  
 **for** (Integer x = 0; x < xAvgList.size(); x++) {  
 **if** (fieldTable[y][x] != **null**) {  
 pageText += fieldTable[y][x].getText() + **"\n"**;  
 *// 去除换行开头空格* fieldTable[y][x].setValueStr(fieldTable[y][x].getValueStr().trim().replaceAll(**"\n[ ]+"**, **"\n"**));  
 }  
 **if** (fieldTable[y][x] != **null** && fieldTable[y][x].getFinish() && fieldTable[y][x].getType().equals(**"KV"**)) {  
 System.***out***.println(**"关键字:"** + fieldTable[y][x].getKeyStr());  
 System.***out***.println(**"内容:"** + fieldTable[y][x].getValueStr());  
 dataMap.put(fieldMap.get(fieldTable[y][x].getFieldId()).getFieldCode(), fieldTable[y][x]);  
 }  
 }  
 }  
 **for** (Map.Entry<Integer, OcrFieldDTO> fieldEntry : fieldMap.entrySet()) { *// 遍历所有字段记录* OcrFieldDTO fieldDTO = fieldEntry.getValue();  
 **if** (fieldDTO.getDirectSearchFlag() && StringUtil.*isNotEmpty*(fieldDTO.getDirectSearchReg())) { *// 如果可以直接搜索并且正则不为空* FieldTableCellDto fieldCellDto = dataMap.get(fieldDTO.getFieldCode());  
 **if** (dataMap.get(fieldDTO.getFieldCode()) != **null** && StringUtil.*isNotEmpty*(fieldCellDto.getValueStr())) { *// 对已匹配数据校验* **if** (fieldDTO.getRecogType() == ***RECOG\_TYPE\_REG***) {  
 *// 思路: 如果已匹配数据中存在正则直接匹配的结果，则判定匹配成功，如果不存在则执行全文检索，如果全文检索中存在匹配结果，则覆盖原结果* String regResultStr = getRegStr(fieldCellDto.getValueStr(), fieldDTO.getDirectSearchReg());  
 **if** (regResultStr.length() == 0) { *//没有匹配到结果 执行全文检索* String pageRegResultStr = getRegStr(pageText, fieldDTO.getDirectSearchReg());  
 **if** (StringUtil.*isNotEmpty*(pageRegResultStr)) {  
 fieldCellDto.setValueStr(pageRegResultStr);  
 }  
 }  
 String valueStr = fieldCellDto.getValueStr();  
 **if** (StringUtil.*isNotEmpty*(valueStr)) { *// 全文检索匹配到结果 重置该字段  
 // 拆分数据  
 //* ***TODO 多个结果展示*** *// 对结果进行拆分 2018-10-15* **if** (StringUtil.*isNotEmpty*(fieldDTO.getValueFormula())) {  
 Object object = **this**.valueFormula(valueStr.trim(), fieldDTO.getValueFormula());  
 **if** (object **instanceof** String) {  
 fieldCellDto.setValueStr((String) object);  
 }  
 }  
 *// 全文检索匹配到结果* }  
 } **else if** (fieldDTO.getRecogType() == ***RECOG\_TYPE\_REG\_PRE\_MAP***) {  
 *// 第一次正则模糊匹配* List<String> regStrArray = getRegStrArray(fieldCellDto.getValueStr(), fieldDTO.getDirectSearchReg());  
 String resultStr = **""**;  
 Map<String, String> resultCodeMap = **new** LinkedHashMap<>(); *// key：业务代码 value：带业务代码的字符串* **if** (**matchingMap**.get(fieldDTO.getRecogStrategy()) != **null**) {  
 Map<String, String> checkMap = **matchingMap**.get(fieldDTO.getRecogStrategy());  
 **if** (regStrArray.size() > 0) {  
 **for** (String regItemStr : regStrArray) {  
 **if** (fieldDTO.getPreType() == 1) { *// 预处理方式:正则  
 // 预处理结果(二次正则)* String preRegStr = getRegStr(regItemStr, fieldDTO.getPreStrategy());  
 **if** (StringUtil.*isNotEmpty*(preRegStr) && checkMap.get(preRegStr) != **null**) { *// 匹配成功* String resultCodeStr = resultCodeMap.get(preRegStr); *// 定义结果Map* **if** (resultCodeStr != **null**) {  
 **if** (regItemStr.trim().length() > resultCodeStr.length()) {  
 resultCodeMap.put(preRegStr, regItemStr.trim()); *// 将更长的结果放入Map* }  
 } **else** {  
 resultCodeMap.put(preRegStr, regItemStr.trim());  
 }  
 }  
 } **else if** (fieldDTO.getPreType() == 2) { *// 预处理方式:Map  
 //* ***TODO*** }  
 }  
 List<Map<String, String>> dataList = **new** ArrayList<>();  
 **for** (Map.Entry<String, String> resultCodeEntry : resultCodeMap.entrySet()) {  
 *//* ***TODO 多个结果展示*** *// 对结果进行拆分 2018-10-15* String regPageStr = resultCodeEntry.getValue();  
 **if** (StringUtil.*isNotEmpty*(fieldDTO.getValueFormula())) {  
 Object object = **this**.valueFormula(resultCodeEntry.getValue().trim(), fieldDTO.getValueFormula());  
 **if** (object **instanceof** String) {  
 **if** (resultStr.length() < ((String) object).length()) {  
 resultStr = (String) object;  
 }  
 } **else if** (object **instanceof** Map) {  
 dataList.add((Map<String, String>) object);  
 }  
 } **else** {  
 **if** (resultStr.length() < regPageStr.length()) {  
 resultStr = regPageStr;  
 }  
 }  
 }  
 *// 全文检索匹配到结果* fieldCellDto.setValueStr(resultStr.trim());  
 **if** (dataList.size() > 0) {  
 fieldCellDto.setDataList(dataList);  
 }  
 }  
 **if** (regStrArray == **null** || resultStr.length() == 0) { *//没有模糊匹配到结果 或者在fieldValue中没有匹配到结果 执行全文检索* List<String> regPageStrArray = getRegStrArray(pageText, fieldDTO.getDirectSearchReg());  
 String pageRegResultStr = **""**;  
 **if** (regPageStrArray.size() > 0) {  
 **for** (String regPageItemStr : regPageStrArray) {  
 **if** (fieldDTO.getPreType() == 1) { *// 预处理方式:正则  
 // 预处理结果(二次正则)* String preRegPageStr = getRegStr(regPageItemStr, fieldDTO.getPreStrategy());  
 String businessCode = checkMap.get(preRegPageStr);  
 **if** (StringUtil.*isNotEmpty*(preRegPageStr) && businessCode != **null**) { *// 匹配成功* String resultCodeStr = resultCodeMap.get(preRegPageStr); *// 定义结果Map* **if** (resultCodeStr != **null**) {  
 **if** (regPageItemStr.trim().length() > resultCodeStr.length()) {  
 resultCodeMap.put(preRegPageStr, regPageItemStr.trim()); *// 将更长的结果放入Map* }  
 } **else** {  
 resultCodeMap.put(preRegPageStr, regPageItemStr.trim());  
 }  
 }  
 } **else if** (fieldDTO.getPreType() == 2) { *// 预处理方式:Map  
 //* ***TODO*** }  
 }  
 List<Map<String, String>> dataList = **new** ArrayList<>();  
 **for** (Map.Entry<String, String> resultCodeEntry : resultCodeMap.entrySet()) {  
 *//* ***TODO 多个结果展示*** *// 对结果进行拆分 2018-10-15* String regPageStr = resultCodeEntry.getValue();  
 **if** (StringUtil.*isNotEmpty*(fieldDTO.getValueFormula())) {  
 Object object = **this**.valueFormula(resultCodeEntry.getValue().trim(), fieldDTO.getValueFormula());  
 **if** (object **instanceof** String) {  
 **if** (resultStr.length() < ((String) object).length()) {  
 resultStr = (String) object;  
 }  
 } **else if** (object **instanceof** Map) {  
 dataList.add((Map<String, String>) object);  
 }  
 } **else** {  
 **if** (resultStr.length() < regPageStr.length()) {  
 resultStr = regPageStr;  
 }  
 }  
 }  
 *// 全文检索匹配到结果* fieldCellDto.setValueStr(resultStr.trim());  
 **if** (dataList.size() > 0) {  
 fieldCellDto.setDataList(dataList);  
 }  
 }  
  
 }  
 }  
 }  
  
 } **else** {  
 **if** (fieldDTO.getRecogType() == ***RECOG\_TYPE\_REG***) {  
 *// 全文检索匹配到结果 创建新字段* List<String> regPageStrArray = getRegStrArray(pageText, fieldDTO.getDirectSearchReg());  
 String resultStr = **""**;  
 List<Map<String, String>> dataList = **new** ArrayList<>();  
 *//* ***TODO 多个结果展示*** *// 对结果进行拆分 2018-10-15* **for** (String regPageStr : regPageStrArray) {  
 **if** (StringUtil.*isNotEmpty*(fieldDTO.getValueFormula())) {  
 Object object = **this**.valueFormula(regPageStr.trim(), fieldDTO.getValueFormula());  
 **if** (object **instanceof** String) {  
 **if** (resultStr.length() < ((String) object).length()) {  
 resultStr = (String) object;  
 }  
 } **else if** (object **instanceof** Map) {  
 dataList.add((Map<String, String>) object);  
 }  
 } **else** {  
 **if** (resultStr.length() < regPageStr.length()) {  
 resultStr = regPageStr;  
 }  
 }  
 }  
 FieldTableCellDto newFieldCellDto = **new** FieldTableCellDto();  
 newFieldCellDto.setValueStr(resultStr.trim());  
 **if** (dataList.size() > 0) {  
 newFieldCellDto.setDataList(dataList);  
 }  
 dataMap.put(fieldDTO.getFieldCode(), newFieldCellDto);  
 } **else if** (fieldDTO.getRecogType() == ***RECOG\_TYPE\_REG\_PRE\_MAP***) {  
 *// 第一次正则模糊匹配* List<String> regStrArray = getRegStrArray(pageText, fieldDTO.getDirectSearchReg());  
 String resultStr = **""**;  
 Map<String, String> resultCodeMap = **new** LinkedHashMap<>(); *// key：业务代码 value：带业务代码的字符串* **if** (regStrArray != **null** && **matchingMap**.get(fieldDTO.getRecogStrategy()) != **null**) {  
 Map<String, String> checkMap = **matchingMap**.get(fieldDTO.getRecogStrategy());  
 **for** (String regItemStr : regStrArray) {  
 **if** (fieldDTO.getPreType() == 1) { *// 预处理方式:正则  
 // 预处理结果(二次正则)* String preRegStr = getRegStr(regItemStr, fieldDTO.getPreStrategy());  
 String businessCode = checkMap.get(preRegStr.toUpperCase());  
 **if** (StringUtil.*isNotEmpty*(preRegStr) && businessCode != **null**) { *// 匹配成功* String resultCodeStr = resultCodeMap.get(preRegStr); *// 定义结果Map* **if** (resultCodeStr != **null**) {  
 **if** (preRegStr.trim().length() > resultCodeStr.length()) {  
 resultCodeMap.put(preRegStr, regItemStr.trim()); *// 将更长的结果放入Map* }  
 } **else** {  
 resultCodeMap.put(preRegStr, regItemStr.trim());  
 }  
 }  
 } **else if** (fieldDTO.getPreType() == 2) { *// 预处理方式:Map  
 //* ***TODO*** }  
 }  
 List<Map<String, String>> dataList = **new** ArrayList<>();  
 **for** (Map.Entry<String, String> resultCodeEntry : resultCodeMap.entrySet()) {  
 *//* ***TODO 多个结果展示*** *// 对结果进行拆分 2018-10-15* String regPageStr = resultCodeEntry.getValue();  
 **if** (StringUtil.*isNotEmpty*(fieldDTO.getValueFormula())) {  
 Object object = **this**.valueFormula(resultCodeEntry.getValue().trim(), fieldDTO.getValueFormula());  
 **if** (object **instanceof** String) {  
 **if** (resultStr.length() < ((String) object).length()) {  
 resultStr = (String) object;  
 }  
 } **else if** (object **instanceof** Map) {  
 dataList.add((Map<String, String>) object);  
 }  
 } **else** {  
 **if** (resultStr.length() < regPageStr.length()) {  
 resultStr = regPageStr;  
 }  
 }  
 }  
 *// 全文检索匹配到结果 创建新字段* FieldTableCellDto newFieldCellDto = **new** FieldTableCellDto();  
 newFieldCellDto.setValueStr(resultStr.trim());  
 **if** (dataList.size() > 0) {  
 newFieldCellDto.setDataList(dataList);  
 }  
 dataMap.put(fieldDTO.getFieldCode(), newFieldCellDto);  
 }  
 }  
  
 }  
 } **else** {  
 FieldTableCellDto fieldCellDto = dataMap.get(fieldDTO.getFieldCode());  
 **if** (StringUtil.*isNotEmpty*(fieldDTO.getValueFormula())) {  
 Object object = **this**.valueFormula(fieldCellDto.getValueStr().trim(), fieldDTO.getValueFormula());  
 **if** (object **instanceof** String) {  
 fieldCellDto.setValueStr((String) object);  
 }  
 }  
 }  
 }  
 *// 处理收发通 拆分* dataMap = splitResultField(dataMap);  
 ocrResultDataDTO.setFieldData(dataMap);  
 System.***out***.println(**"结束处理结果:"** + **new** Date().getTime());  
 **return** ocrResultDataDTO;  
 }  
  
  
 **private** List<FieldTableCellDto> cellDataSplit(FieldTableCellDto fieldTableCellDto) {  
 List<FieldTableCellDto> fieldList = **new** ArrayList<>();  
 FieldTableCellDto cellDto = (FieldTableCellDto) fieldTableCellDto.clone();  
 List<FieldSplitDto> fieldSplitDtoList = cellDto.getFieldSplitList();  
 Collections.*sort*(cellDto.getFieldSplitList(), (arg0, arg1) -> {  
 **int** i = arg0.getIndex().compareTo(arg1.getIndex());  
 **return** i;  
 });  
 Integer totalLine = *getLineCount*(fieldTableCellDto.getText());  
 String text = cellDto.getText(); *// 定义要拆分的文本内容* Integer cursorPos = 0; *// 定义光标位置* FieldTableCellDto baseCell = **new** FieldTableCellDto();  
 baseCell.setFieldId(cellDto.getFieldId());  
 baseCell.setTop(cellDto.getTop());  
 baseCell.setBottom(cellDto.getBottom());  
 baseCell.setLeft(cellDto.getLeft());  
 baseCell.setRight(cellDto.getRight());  
 baseCell.setText(text.substring(0, fieldSplitDtoList.get(0).getIndex())); *//切断  
 //识别数据 可优化* String[] baseSplitResult = *splitByStr*(baseCell.getText(), fieldTableCellDto.getKeyStr());  
 **if** (!fieldTableCellDto.getType().equals(**"V"**)) { *// 判断如果单元不是“值”，则执行拆分* **if** (baseSplitResult.**length** > 1) {  
 baseCell.setKeyStr(fieldTableCellDto.getKeyStr());  
 baseCell.setValueStr(baseSplitResult[1].trim());  
 baseCell.setType(**"KV"**); *// 单元格包含标题和值* baseCell.setFinish(**true**);  
 } **else** {  
 baseCell.setKeyStr(fieldTableCellDto.getKeyStr());  
 baseCell.setType(**"K"**); *// 单元格只包含标题* }  
 } **else** {  
 baseCell.setValueStr(baseCell.getText().trim()); *// 如果是"V" 则直接数出内容作为结果.* baseCell.setType(**"V"**); *// 单元格为数据或未识别内容* }  
 fieldList.add(baseCell);  
 LinkedHashMap<Integer, List<FieldSplitDto>> splitMap = **new** LinkedHashMap<>();  
 Integer baseCellHeight = baseCell.getBottom() - baseCell.getTop();  
 Integer baseCellWidth = baseCell.getLeft() - baseCell.getRight();  
  
 Boolean rowSplitFlag = **false**; *// 横向拆分标志，用于检测是否整列拆分,如果第一行没有检测到换行并且有多个关键字 则整列拆分* Boolean firstRowSplitFinish = **false**; *// 纵向拆分第一行* Integer rows = 0;  
 **for** (**int** index = 0; index < fieldSplitDtoList.size(); index++) {  
 FieldSplitDto fieldSplitDto = fieldSplitDtoList.get(index);  
 Integer cursorLine = *getLineCount*(text.substring(0, fieldSplitDto.getIndex())); *// 操作第几行* **if** (splitMap.get(cursorLine) == **null**) {  
 List<FieldSplitDto> fieldSplitList = **new** ArrayList<>();  
 splitMap.put(cursorLine, fieldSplitList);  
 }  
 **if** (fieldSplitDto.getSplitDirection() == **"Y"**) {  
 rows++;  
 }  
 **if** (index + 1 < fieldSplitDtoList.size()) { *// 不是最后一个* fieldSplitDto.setEndIndex(fieldSplitDtoList.get(index + 1).getIndex());  
 } **else** {  
 fieldSplitDto.setEndIndex(text.length());  
 }  
 splitMap.get(cursorLine).add(fieldSplitDto);  
 }  
  
 **for** (Map.Entry<Integer, List<FieldSplitDto>> splitListEntry : splitMap.entrySet()) {  
 **int** y = splitListEntry.getKey();  
 List<FieldSplitDto> fieldSplitList = splitListEntry.getValue();  
 **int** cols = fieldSplitList.size();  
 **for** (**int** x = 0; x < cols; x++) {  
 FieldSplitDto item = fieldSplitList.get(x);  
 FieldTableCellDto fieldDto = **new** FieldTableCellDto();  
 fieldDto.setText(text.substring(item.getIndex(), item.getEndIndex()));  
 fieldDto.setFieldId(item.getFieldId()); *// 设置fieldId  
 //识别数据 可优化* String[] result = *splitByStr*(fieldDto.getText(), item.getFieldStr());  
 **if** (result.**length** > 1) {  
 fieldDto.setKeyStr(item.getFieldStr());  
 fieldDto.setValueStr(result[1].trim());  
 fieldDto.setType(**"KV"**); *// 单元格包含标题和值* fieldDto.setFinish(**true**);  
 } **else** {  
 fieldDto.setKeyStr(item.getFieldStr());  
 fieldDto.setType(**"K"**); *// 单元格只包含标题* }  
 **if** (x == 0) {  
 fieldDto.setLeft(baseCell.getLeft());  
 } **else** {  
 fieldDto.setLeft((**int**) Math.*round*(((**double**) (x) / cols) \* baseCellWidth));  
 }  
 **if** (x == cols - 1) {  
 fieldDto.setRight(baseCellWidth);  
 } **else** {  
 fieldDto.setRight((**int**) Math.*round*(((**double**) (x + 1) / cols) \* baseCellWidth));  
 }  
 **if** (y == 1) {  
 fieldDto.setTop(baseCell.getTop());  
 } **else** {  
 fieldDto.setTop((**int**) Math.*round*(((**double**) (y) / totalLine) \* baseCellHeight));  
 }  
 **if** (y == rows) {  
 fieldDto.setBottom(baseCellHeight);  
 } **else** {  
 fieldDto.setBottom((**int**) Math.*round*(((**double**) (y + *getStartEnterStrCount*(fieldDto.getText())) / totalLine) \* baseCellHeight));  
 }  
 fieldList.add(fieldDto);  
 }  
 }  
  
 **return** fieldList;  
 }  
  
 **private** OcrResultDataDTO processImage(String filePath) {  
*// String imagePath = SamplesConfig.GetSamplesFolder() + "\\SampleImages\\Demo.tif";* String imagePath = filePath;  
 **try** {  
 *// Don't recognize PDF file with a textual content, just copy it  
 /\* if( engine.IsPdfWithTextualContent( imagePath, null ) ) {  
 displayMessage( "Copy results..." );  
 String resultPath = SystemConfig.GetSamplesFolder() + "\\rcs.rtf";  
 Files.copy( Paths.get( imagePath ), Paths.get( resultPath ), StandardCopyOption.REPLACE\_EXISTING );  
 return;  
 }\*/  
  
 // Create document* IFRDocument document = **engine**.CreateFRDocument();  
 **try** {  
 System.***out***.println(**"OCR开始时间:"** + **new** Date().getTime());  
 *// Add image file to document  
 displayMessage*(**"Loading image..."**);  
 document.AddImageFile(imagePath, **null**, **null**);  
  
 *// Process document  
 displayMessage*(**"Process..."**);  
 IDocumentProcessingParams processingParams = **engine**.CreateDocumentProcessingParams();  
 IRecognizerParams recognizerParams = processingParams.getPageProcessingParams().getRecognizerParams();  
 recognizerParams.SetPredefinedTextLanguage(**"ChinesePRC, English"**);  
 document.Process(processingParams);  
*// document.Process( null );  
  
 // Print the plain text result  
// IPlainText plainText = document.getPlainText();  
// displayMessage(plainText.getText());* IFRPages pages = document.getPages();  
 **int** pagesCount = pages.getCount();  
 System.***out***.println(**"Total page count: "** + pagesCount);  
  
 IFRPage page = pages.Item(0);  
 *// iterate pages – work with blocks, paragraphs, characters* System.***out***.println(**"OCR结束时间:"** + **new** Date().getTime());  
 *// 返回结果* OcrResultDataDTO ocrResultData = **new** OcrResultDataDTO();  
 ocrResultData = iteratePage(page);  
  
 *// if the page was changed in iteratePage(), call Flush(true) to keep changes  
// page.Flush(true);  
 // if there is no need to keep changes, call Flush(false)* page.Flush(**false**);  
  
  
 *// Save results  
 displayMessage*(**"Saving results..."**);  
  
 */\* // Save results to rtf with default parameters  
 String rtfExportPath = SystemConfig.GetSamplesFolder() + "\\Imag.rtf";  
 document.Export(rtfExportPath, FileExportFormatEnum.FEF\_RTF, null);  
  
 // Save results to pdf using 'balanced' scenario  
 IPDFExportParams pdfParams = engine.CreatePDFExportParams();  
 pdfParams.setScenario(PDFExportScenarioEnum.PES\_Balanced);  
  
 String pdfExportPath = SystemConfig.GetSamplesFolder() + "\\Imag.pdf";  
 document.Export(pdfExportPath, FileExportFormatEnum.FEF\_PDF, pdfParams);\*/* **return** ocrResultData;  
 } **finally** {  
 *// Close document* document.Close();  
 }  
 } **catch** (Exception ex) {  
 System.***out***.print(ex);  
 }  
 **return null**;  
 }  
  
 **private void** unloadEngine() **throws** Exception {  
 *displayMessage*(**"Deinitializing Engine..."**);  
 **engine** = **null**;  
 Engine.*DeinitializeEngine*();  
 }  
  
 **private static void** displayMessage(String message) {  
 System.***out***.println(message);  
 }  
  
 **private static** Integer getApproximateIndex(List<Integer> list, **int** x) {  
 **if** (list == **null**) {  
 **return** -1;  
 }  
 **if** (list.size() == 1) {  
 **return** list.get(0);  
 }  
 **int** minDifference = Math.*abs*(list.get(0) - x);  
 **int** minIndex = 0;  
 **for** (**int** i = 1; i < list.size(); i++) {  
 **int** temp = Math.*abs*(list.get(i) - x);  
 **if** (temp < minDifference) {  
 minIndex = i;  
 minDifference = temp;  
 }  
 }  
 **return** minIndex;  
 }  
  
 *// 得到坐标接近的最小值* **private static** Integer getBoxAxisDis(List<Integer> list, Integer val) {  
 Integer index = 0;  
 **if** (list.size() > 0) {  
 **for** (**int** i = 0; i < list.size(); i++) {  
 **if** (val - 15 > list.get(i)) {  
 index = i;  
 } **else** {  
 **break**;  
 }  
 }  
 }  
 **return** index;  
 }

import cv2

import numpy as np

src = cv2.imread("E:/BUSAN/BUSAN\_1.png")

gray\_src = cv2.cvtColor(src, cv2.COLOR\_BGR2GRAY)

gray\_src = cv2.bitwise\_not(gray\_src)

binary\_src = cv2.adaptiveThreshold(gray\_src, 255, cv2.ADAPTIVE\_THRESH\_MEAN\_C, cv2.THRESH\_BINARY, 15, -2)

cv2.namedWindow("result image", cv2.WINDOW\_AUTOSIZE)

cv2.imshow("result image", binary\_src)

# 提取水平线

hline = cv2.getStructuringElement(cv2.MORPH\_RECT, ((src.shape[1] // 30), 1), (-1, -1))

# 提取垂直线

vline = cv2.getStructuringElement(cv2.MORPH\_RECT, (1, (src.shape[0] // 70)), (-1, -1))

# 这两步就是形态学的开操作——先腐蚀再膨胀

#图像腐蚀

dstV = cv2.morphologyEx(binary\_src, cv2.MORPH\_OPEN, vline)

dstV = cv2.bitwise\_not(dstV)

dstH = cv2.morphologyEx(binary\_src, cv2.MORPH\_OPEN, hline)

dstH = cv2.bitwise\_not(dstH)

#取交点 取得横竖线交叉点

dstV1 = dstV

cv2.imshow("dst1", dstV)

dstV1 = np.delete(dstV1, 0, axis = 0)

cv2.imshow("dst2", dstV1)

cv2.imshow("dst3", dstH)

def lineExtend(img, axis, dis = 5):

x=np.ones((dis,src.shape[1 if(axis == 0) else 0]))\*255

if axis == 1:

imgNew = np.delete(img, range(0,dis), axis = 0)

imgNew = np.insert(imgNew, src.shape[0] - dis, values=np.ones((dis,src.shape[1]), dtype=int) \* 255, axis=0)

imgNew = np.bitwise\_and(img, imgNew)

cv2.imshow("input image222" + str(1), imgNew)

imgNew2 = np.delete(img, range(src.shape[0] - dis - 1, src.shape[0] - 1), axis = 0)

imgNew2 = np.insert(imgNew2, 0, values=np.ones((dis, src.shape[1]), dtype=int)\*255, axis = 0)

imgNew2 = np.bitwise\_and(img, imgNew2)

cv2.imshow("input image223" + str(1), imgNew2)

else:

imgNew = np.delete(img, range(0, dis), axis=1)

imgNew = np.insert(imgNew, src.shape[1] - dis - 1, values=np.ones((dis,src.shape[0]), dtype=int) \* 255, axis=1)

imgNew = np.bitwise\_and(img, imgNew)

imgNew2 = np.delete(img, range(src.shape[1] - dis - 1, src.shape[1] - 1), axis=1)

imgNew2 = np.insert(imgNew2, 0, values=np.ones((dis,src.shape[0]), dtype=int) \* 255, axis=1)

imgNew2 = np.bitwise\_and(img, imgNew2)

result = np.bitwise\_and(imgNew2, imgNew)

cv2.imshow("input image1151" + str(0), result)

return result

dstVRe = lineExtend(dstV, 1,10);

cv2.imshow("dst41", dstVRe)

dstHRe = lineExtend(dstH, 0, 5);

cv2.imshow("dst5", dstVRe + dstHRe)

cv2.waitKey(0)

#异或操作 作用:通过错位找到横纵线条顶点，求交集

dstVNew = dstV

dstVNew = np.delete(dstVNew, 0, axis = 0)

dstVNew = np.insert(dstVNew, src.shape[0]-1, values = 255 , axis = 0)

dstVRe = np.bitwise\_xor(dstV ,dstVNew)

dstVRe = np.delete(dstVRe, src.shape[0]-1, axis = 0)

dstVRe = np.insert(dstVRe, 0, values = 0 , axis = 0)

dstHNew = dstH

dstHNew = np.delete(dstHNew, 0, axis = 1)

dstHNew = np.insert(dstHNew, src.shape[1]-1, values = 255 , axis = 1)

dstHRe = np.bitwise\_xor(dstH ,dstHNew)

dstHRe = np.delete(dstHRe, src.shape[1]-1, axis = 1)

dstHRe = np.insert(dstHRe, 0, values = 0 , axis = 1)

result = dstHRe + dstVRe

result = np.argwhere(result == 255)

print(result)

cv2.waitKey(0)