**数据挖掘实践任务教学平台**

**源代码**

|  |  |  |
| --- | --- | --- |
| 文件状态：  [ ] 草稿  [√] 正式发布  [ ] 正在修改 | 文件标识： | SourceDoc |
| 当前版本： | 2.1 |
| 作 者： | 刘祥德、马腾、刘鑫容 |
| 完成日期： | 2018.9.9 |

# 前端

## 数据集模块

<template>

<div id="app" class="app-container">

<Row>

<Col span="2">&nbsp;</Col>

<Col span="20">

<div class="details-container">

<div class="title">{{ collectionName }}</div>

<div class="details-container">

<Card >

<div slot="title">

<el-row>

<el-col :span="24">

<Icon type="arrow-shrink"></Icon>

<span class="title-font-style"> 数据集文件 </span>

<Button type="ghost" class="btn-download-sets"

:disabled="!checkedSetIds.length"

:loading="loading"

@click="handleDownloadSetZip">{{ loading ? "正在压缩" : "下载" }}</Button>

<Button type="error"

class="btn-download-sets"

:disabled="!checkedSetIds.length || readOnly"

@click="handleDeleteSets">删除</Button>

</el-col>

</el-row>

</div>

<a href="#" slot="extra"

@click.prevent="uploadDialogVisible = true">

<Icon type="plus-round"></Icon>

新增

</a>

<el-dialog

title="新增数据集"

:visible.sync="uploadDialogVisible"

width="30%">

<el-upload

class="upload-container"

drag

action="Deprecated"

ref="uploadDataSets"

:on-remove="handleFileRemove"

:on-preview="handleFilePreview"

:before-remove="beforeFileRemove"

:before-upload="handleBeforeUpload"

:on-success= "handleUploadSuccess"

:file-list="fileList"

:auto-upload="false"

multiple>

<i class="el-icon-upload"></i>

<div class="el-upload\_\_text">将文件拖到此处，或<em>点击上传</em></div>

<!--<div class="el-upload\_\_tip" slot="tip">只能上传符合模板的.xlsx文件</div>-->

</el-upload>

<span slot="footer" class="dialog-footer">

<el-button @click="uploadDialogVisible = false">取 消</el-button>

<el-button type="primary" @click="submitUpload">确 定</el-button>

</span>

</el-dialog>

<a href="#" slot="extra" @click.prevent="refreshDataSet()">

<Icon type="ios-loop-strong"></Icon>

刷新

</a>

<CheckboxGroup v-model="checkedSetIds">

<animate-transition :link-index="linkIndex"

:in-style="'transition.bounceLeftIn'"

:out-style="'transition.bounceRightOut'">

<Alert v-for="(item, index) in dataSets"

v-bind:key="item.containerId"

type="warning"

class="file-alert"

:data-index="index">

<Checkbox :label="item.containerId" class="checkbox-container">

{{item.null}}

</Checkbox>

<li>

<a class="file-download-a">{{ item.fileName }}

<a class="file-download-size" >{{ dataSetSizes[index] }}</a>

</a>

</li>

</Alert>

</animate-transition>

</CheckboxGroup>

<Page :total="totalSets"

:page-size="listQuery.size"

@on-change="handlePageChange"

@on-page-size-change="handleSizeChange"

class="page-style"

size="small"

show-elevator

show-total

show-sizer></Page>

</Card>

</div>

<el-row>

<el-col style="margin-left: 40px;margin-bottom: 20px;float: left">

<transition name="fade" mode="out-in">

<Progress v-if="displayProgress" :percent="\_percent"></Progress>

</transition>

</el-col>

</el-row>

<Card :bordered="false" class="dataSetCard">

<p slot="title" style="font-size: 16px">描述摘要:</p>

<p >{{ abstractInfo }}</p>

</Card>

<template>

<h3 class="updown">描述信息:</h3>

<Table :columns="columns1" :data="data1"></Table>

<Table :columns="columns2" :data="data2"></Table>

<Table :columns="columns3" :data="data3"></Table>

</template>

<Card :bordered="false" class="dataSetCard"

v-for="todo in todos"

v-bind:key="todo.id"

v-bind:title="todo.title">

<p slot="title" style="font-size:16px;">{{ todo.title }}</p>

<p >{{ todo.info }}</p>

</Card>

</div>

</Col>

</Row>

</div>

</template>

<script type="text/javascript">

import FileSaver from 'file-saver'

import {

fetchOptions,

getCollection,

getSets,

addSet,

fetchCollectionList,

downloadSetZip,

deleteBatchDataSets

} from 'api/datasets';

export default {

props: {

//只读

readOnly:false,

toCollection:{}

},

name: 'DataSetDetail',

created() {

this.collectionId = this.$route.query.collectionId;

//路由直接进入

if (this.collectionId !== undefined) {

this.getCollectionById(this.collectionId);

this.getDataSets(this.collectionId);

}

//依赖prop传入的collection

else if(this.toCollection.collectionId !== undefined){

this.collection = this.toCollection;

this.collectionId = this.toCollection.collectionId;

this.getDataSets(this.collectionId);

}

else{

this.collection = this.toCollection;

}

},

data() {

return {

uploadDialogVisible: false,

isDataSetRefresh: false,

checkedSetIds: [],

listQuery: {

page: 0,

size: 20,

sort: "containerId,ASC",

},

linkIndex: true,

totalSets: 0,

columns1: [

{

title: '特征',

key: 'dataSetCharacteristics'

},

{

title: '属性类型',

key: 'attributeCharacteristics'

},

{

title: '相关任务',

key: 'associatedTasks'

}

],

columns2: [

{

title: '实例数',

key: 'numberOfInstances'

},

{

title: '允许缺省?',

key: 'isMissingValues'

}

],

columns3: [

{

title: '来自地区',

key: 'area'

},

{

title: '捐赠时间',

key: 'dateDonated'

},

{

title: '点击量',

key: 'numberOfWebHits'

}

],

todos: [

{

id: 1,

title: '来源:',

info: 'Song Xi Chen, csx \'@\' gsm.pku.edu.cn, Guanghua School of Management, ' +

'Center for Statistical Science, Peking University.'

},

{

id: 4,

title: '相关论文:',

info: 'Liang, X., S. Li, S. Zhang, H. Huang, and S. X. Chen (2016), PM2.5 data reliability, ' +

'consistency, and air quality assessment in five Chinese cities, J. Geophys. Res. Atmos., 121, 10220â€“10236.'

},

],

collection: {

collectionId: '',

collectionName: '',

area: '',

associatedTasks: [],

attributeCharacteristics: [],

dataSetCharacteristics: [],

dataSetFolderPath: '',

descriptions: [],

isMissingValues: '',

numberOfInstances: 0

},

dataSets: [],

fileList: [],

loaded:0,

total: 1,

loading: false,

displayProgress:false

};

},

methods: {

async handleBeforeUpload(file) {

//新建一个Form data 类型的文件

let fd = new FormData();

fd.append('file', file);

//发起上传请求

//生成一个上传进度查询key

const uuid\_v1 = require('uuid/v1');

let progress\_uuid = uuid\_v1();

let vm = this;

addSet(this.collectionId, fd, progress\_uuid).then((res) => {

this.$message({

message: '上传成功',

type: 'info',

duration: 1500

});

vm.uploadDialogVisible = false;

vm.refreshDataSet();

}).catch(error => {

console.log(error);

});

},

handleUploadSuccess(response, file, fileList) {

this.uploadDialogVisible = false;

this.$message({

message: '上传成功',

type: 'info',

duration: 1500

});

},

handleDownloadEvent(event){

this.displayProgress = true;

this.loaded = event.loaded;

this.total = event.total;

if(event.loaded === event.total){

this.displayProgress = false;

}

},

handleFileChange(file, fileList) {

this.fileList = fileList;

},

handleFileRemove(file, fileList) {

console.log(file, fileList);

},

handleFilePreview(file) {

console.log(file.response);

},

beforeFileRemove(file, fileList) {

return this.$confirm(`确定移除 ${ file.name }？`);

},

handleSelectionChange(val) {

this.multipleSelection = val;

},

submitUpload() {

this.$refs.uploadDataSets.submit();

},

handleDownloadSetZip() {

let vm = this;

this.loading = true;

downloadSetZip(this.collectionId, this.checkedSetIds,this.handleDownloadEvent).then((res) => {

const effectiveFileName = res.headers['x-suggested-filename'];

vm.loading = false;

FileSaver.saveAs(res.data, effectiveFileName);

}).catch((res) => {

vm.loading = false;

console.error("Could not download the zip files from the backend.", res);

});

},

handleSizeChange(val) {

this.listQuery.size = val;

this.getDataSets(this.collectionId);

},

handlePageChange(val) {

this.listQuery.page = val - 1;

this.getDataSets(this.collectionId);

},

checkedContainers() {

let vm = this;

let checkedContainers = [];

this.checkedSetIds.forEach(c => {

vm.dataSets.forEach(d => {

if (d.containerId === c) {

checkedContainers.push(d);

}

})

});

return checkedContainers;

},

handleDeleteSets() {

let vm = this;

this.$confirm('此操作将删除对应的数据集文件, 是否继续?', '确定删除', {

confirmButtonText: '确定',

cancelButtonText: '取消',

type: 'warning'

}).then(() => {

deleteBatchDataSets(this.checkedSetIds).then(() => {

vm.$message({

message: '删除成功',

type: 'success',

duration: 1500

});

let checked = vm.checkedContainers();

checked.forEach(s => {

let index = vm.dataSets.indexOf(s);

vm.dataSets.splice(index, 1);

});

vm.checkedSetIds = [];

}).catch(error => {

});

}).catch(() => {

vm.$message({

type: 'info',

message: '取消删除'

});

});

},

refreshDataSet() {

this.getDataSets(this.collectionId);

},

getCollectionById(collectionId) {

let vm = this;

this.loading = true;

getCollection(collectionId).then(response => {

vm.collection = response;

vm.loading = false;

}).catch(error => {

console.log(error);

})

},

getDataSets(collectionId) {

let vm = this;

vm.linkIndex = true;

this.isDataSetRefresh = true;

getSets(collectionId, Object.assign({}, this.listQuery)).then(res => {

vm.dataSets = res.content;

vm.totalSets = res.totalElements;

this.isDataSetRefresh = false;

})

},

downloadDataSet(containerId) {

}

},

computed: {

\_percent(){

return Math.trunc(this.loaded / this.total \* 100);

},

collectionName() {

return this.collection.collectionName;

},

fixPage() {

return this.listQuery.page + 1;

},

dataSetSizes() {

var numeral = require('numeral');

numeral.defaultFormat('0.00');

return this.dataSets.map(d => {

return numeral((d.size) / (1024 \* 1024)).format() + ' MB';

});

},

abstractInfo() {

return this.collection.abstractInfo;

},

data1() {

return [

{

dataSetCharacteristics: this.collection.dataSetCharacteristics.map(char => char.englishName).join(),

attributeCharacteristics: this.collection.attributeCharacteristics.map(attr => attr.englishName).join(),

associatedTasks: this.collection.associatedTasks.map(task => task.englishName).join()

}

]

},

data2() {

return [{

numberOfInstances: this.collection.numberOfInstances,

isMissingValues: this.collection.isMissingValues

}]

},

data3() {

return [

{

area: this.collection.area.englishName,

dateDonated: this.collection.dateDonated,

numberOfWebHits: this.collection.numberOfWebHits

}

]

},

}

};

</script>

<style scoped="scss">

div{

background-color:transparent;

margin:0px auto;

font-size:16px;

}

.title{

margin-top: 20px;

margin-bottom: 20px;

font:bold 36px 微软雅黑;

text-align: center;

}

.updown{

margin-top: 20px;

margin-bottom: 20px;

font-size:16px;

}

.details-container{

margin: 40px;

}

.title-font-style{

font-weight:bold;

}

/\*.data-set-file-container {\*/

/\*margin-top: 40px;\*/

/\*}\*/

.file-download-a{

background: 0 0;

font-style: italic;

text-decoration: none;

cursor: pointer;

transition: color .2s ease;

}

.page-style{

margin-top: 50px;

}

.file-alert{

margin-top: 3px;

border: 1px solid #d5e8fc;

background-color: #eaf4fe;

}

.file-download-size{

color: #ffac2d;

position: absolute;

right:10%;

}

.btn-download-sets{

margin-left: 10px;

}

.checkbox-container{

position: absolute;

right: 0;

}

.data-set-file-spin-container{

position: absolute;

top: 0;

left: 0;

z-index: 8;

background-color: hsla(0,0%,100%,.9);

}

.dataSetCard{

background-color: transparent;

font-size:16px;

}

.list-complete-item {

transition: all 1s;

display: inline-block;

}

.list-complete-enter, .list-complete-leave-to

/\* .list-complete-leave-active for below version 2.1.8 \*/ {

opacity: 0;

transform: translateY(30px);

}

.list-complete-leave-active {

position: absolute;

}

</style>

## 学生模块

<template>

<div class="calendar-list-container test">

<div class="title">

学生信息管理

</div>

<div class="btn-import-container">

<el-input @keyup.enter.native="handleFilter" style="width: 200px;" @change='handleFilter' class="btn-item" placeholder="学号" v-model="listQuery.studentId">

</el-input>

<el-select clearable style="width: 100px" @change='handleFilter' class="btn-item" v-model="listQuery.grade" placeholder="年级">

<el-option v-for="item in gradeOptions" :key="item" :label="item" :value="item">

</el-option>

</el-select>

<el-select clearable class="btn-item" @change='handleFilter' style="width: 130px" v-model="listQuery.className" placeholder="班级">

<el-option v-for="item in classNameOptions" :key="item.key" :label="item" :value="item">

</el-option>

</el-select>

<el-select @change='handleFilter' style="width: 120px" class="btn-item" v-model="listQuery.sort" placeholder="排序">

<el-option v-for="item in sortOptions" :key="item.key" :label="item.label" :value="item.key">

</el-option>

</el-select>

<el-button class="btn-item" type="primary" v-waves icon="el-icon-search" @click="handleFilter">筛选</el-button>

<el-button class="btn-item" style="margin-left: 10px;" @click="handleCreate" type="primary" icon="el-icon-plus">添加</el-button>

<el-button class="btn-item" type="primary" icon="el-icon-document" @click="handleDownload">导出</el-button>

<el-button class="btn-item" type="primary" icon="el-icon-delete" v-waves @click="handleBatchDelete" :disabled="!multipleSelection.length">批量删除</el-button>

<!--<el-button class="btn-item" type="warning" icon="el-icon-star-on" v-waves @click="markFavoriteStudent" v-if="multipleSelection.length" >收藏</el-button>-->

<!--<el-button class="btn-item" type="warning" icon="el-icon-star-off" :plain="true" v-waves @click="unMarkFavoriteStudent" v-if="multipleSelection.length" >取消收藏</el-button>-->

</div>

<Table border size='default'

:loading="listLoading"

:columns="studentColumns"

:data="studentList"

@on-selection-change="handleSelectionChange"

class="student-list-container"

stripe></Table>

<div class="pagination-container">

<el-pagination @size-change="handleSizeChange" @current-change="handleCurrentChange"

:page-sizes="[10,20,30, 50]"

:page-size="listQuery.size"

background

layout="total, sizes, prev, pager, next, jumper"

:total="total">

</el-pagination>

</div>

<el-dialog :title="textMap[dialogStatus]" :visible.sync="dialogFormVisible">

<el-form class="small-space" :model="temp" label-position="left" label-width="70px" style='width: 400px; margin-left:50px;'>

<el-form-item label="学号" label-width="85px">

<el-input v-model="temp.studentId">

</el-input>

</el-form-item>

<el-form-item label="姓名" label-width="85px">

<el-input v-model="temp.studentName">

</el-input>

</el-form-item>

<el-form-item label="年级" label-width="85px">

<el-select class="btn-item" v-model="temp.grade">

<el-option v-for="item in gradeOptions" :key="item" :label="item" :value="item">

</el-option>

</el-select>

</el-form-item>

<el-form-item label="专业" label-width="85px">

<el-select class="btn-item" v-model="temp.profession">

<el-option v-for="item in professionOptions" :key="item" :label="item" :value="item">

</el-option>

</el-select>

</el-form-item>

<el-form-item label="班级" label-width="85px">

<el-select class="btn-item" v-model="temp.className">

<el-option v-for="item in classNameOptions" :key="item" :label="item" :value="item">

</el-option>

</el-select>

</el-form-item>

<el-form-item label="参与任务数" label-width="85px">

<el-input v-model="temp.finishedTaskCount" :disabled="dialogStatus!=='create'">

</el-input>

</el-form-item>

</el-form>

<div slot="footer" class="dialog-footer">

<el-button @click="dialogFormVisible = false">取 消</el-button>

<el-button v-if="dialogStatus=='create'" type="primary" @click="create">确 定</el-button>

<el-button v-else type="primary" @click="update">确 定</el-button>

</div>

</el-dialog>

</div>

</template>

<script>

import { fetchList, fetchPv } from 'api/article\_table';

import { parseTime,deleteEmptyProperty } from 'utils';

import { fetchStudentList,

deleteStudent ,

createStudent,

updateStudent,

deleteStudentBatch,

markStudent,

unMakrStuden,

fetchOptions

} from 'api/students';

export default {

name: 'StudentTable',

data() {

return {

studentColumns: [

{

type: 'selection',

align:'center',

width: 60,

},

{

title: '序号',

type: 'index',

width: 70,

align: 'center'

},

{

title: '学号',

align: 'center',

key: 'studentId'

},

{

title: '姓名',

align: 'center',

key: 'studentName'

},

{

title: '年级',

align: 'center',

key: 'grade'

},

{

title: '专业',

align: 'center',

key: 'profession'

},

{

title: '班级',

align: 'center',

key: 'className',

},

{

title: '管理',

align: 'center',

width: 200,

render: (h, params) => {

return h('ButtonGroup', [

h('Button', {

props: {

icon:'ios-search',

size:'small'

},

on: {

click: () => {

this.handleCheck(params.index)

}

}

}),

h('Button', {

props: {

icon:'android-delete',

size:'small'

},

on: {

click: () => {

this.handleDelete(params.index)

}

}

}),

h('Button', {

props: {

icon:'edit',

size:'small'

},

on: {

click: () => {

this.handleUpdate(params.index)

}

}

})

]);

}

}

],

total: null,

listLoading: false,

listQuery: {

studentId: "",

studentName: "",

className: "",

profession:"",

grade:"",

page: 0,

size: 20,

sort:"studentId,ASC",

},

temp: {

studentId:'',

studentName:'',

grade:'',

className:'',

profession:'',

status:{

statusId:3,

chineseValue:'空闲',

englishValue:'available',

},

favorite:{

favoriteId:0,

chineseValue:'已收藏',

englishValue:'favorite'

},

finishedTaskCount:0

},

studentList: [],

queryTypeOptions:[

'姓名',

'学号'

],

statusOptions:[{

statusId:0,

chineseValue:'任务进行中',

englishValue:'executing',

},{

statusId:1,

chineseValue:'锁定',

englishValue:'unavailable',

},{

statusId:2,

chineseValue:'任务完成',

englishValue:'finished',

},{

statusId:3,

chineseValue:'空闲',

englishValue:'available',

}],

sortOptions: [{label: '按学号升序', key: 'studentId,ASC'}, {label: '按学号降序', key: 'studentId,DESC'}],

multipleSelection:[],

isDisplayFavoriteColumn:false,

gradeOptions:[],

professionOptions:[],

classNameOptions:[],

dialogFormVisible: false,

dialogStatus: '',

textMap: {

update: '编辑',

create: '创建'

},

dialogPvVisible: false,

pvData: [],

tableKey: 0

};

},

created() {

this.getStudentList();

this.getOptions();

},

computed: {

fixPage(){

return this.listQuery.page + 1;

}

},

methods: {

handleSelectionChange(val) {

this.multipleSelection = val;

console.log(val);

},

handleRowClicked(row,event,column) {

let lable = column.label;

if(lable==='操作'||lable==='算法配置'){

return;

}

this.$refs.studentTable.toggleRowSelection(row);

this.isDisplayFavoriteColumn = !this.isDisplayFavoriteColumn;

},

getOptions() {

let vm = this;

fetchOptions().then(res =>{

vm.gradeOptions = res.gradeOptions;

vm.classNameOptions = res.classNameOptions;

vm.professionOptions = res.gradeOptions;

})

},

getStudentList() {

let that = this;

this.listLoading = true;

fetchStudentList(Object.assign({}, this.listQuery)).then(response => {

this.studentList = response.content;

this.total = response.totalElements;

this.listLoading = false;

}).catch(error =>{

that.$message({

type: 'error',

message:error

})

})

},

handleImport() {

},

handleFilter() {

this.getStudentList();

},

handleSizeChange(val) {

if (this.listQuery.size === val) {

return

}

this.listQuery.size = val;

this.getStudentList();

},

handleCurrentChange(val) {

if (this.listQuery.page === val -1) {

return

}

this.listQuery.page = val -1;

this.getStudentList();

},

handleBatchDelete() {

let confirmMessage = '您将删除所有被选择学生的信息,是否继续?';

let studentIds = this.multipleSelection.map(item => item.studentId);

let that =this;

this.$confirm(confirmMessage,'批量删除学生',{

confirmButtonText:'确认',

cancelButtonText:'取消',

beforeClose: (action,instance,done) =>{

if(action ==='confirm'){

//显示加载按钮

instance.confirmButtonLoading = true;

return new Promise((resolve,reject) =>{

//通过API发送批量删除请求

deleteStudentBatch(studentIds).then(response =>{

instance.confirmButtonLoading=false;

resolve(response);

done();

}).catch(error =>{

//捕获错误;

instance.confirmButtonLoading=false;

reject(error);

done();

})

})

}

//关闭确认框

done();

}

}).then((message) =>{

//删除被选中的行

that.studentList = that.studentList.filter(student =>

!that.multipleSelection.some(row => row.studentId === student.studentId));

// that.getStudentList();

that.$message({

message: '批量删除成功',

type: 'success',

duration: 1500

});

that.getStudentList();

}).catch(() =>{

this.$message({

message: '取消批量删除操作',

type: 'info',

duration: 1500

});

})

},

markFavoriteStudent() {

let favoriteStudents = this.multipleSelection.filter(row => !row.favorite.value);

let studentIds = favoriteStudents.map(student => student.studentId);

let that = this;

markStudent(studentIds).then(response =>{

favoriteStudents.forEach(student => {

student.favorite = {key: '已收藏', value: true};

});

that.$message({

type:'success',

message:'收藏成功'

})

})

},

unMarkFavoriteStudent() {

let favoriteStudents = this.multipleSelection.filter(row => row.favorite.value);

let studentIds = favoriteStudents.map(student => student.studentId);

let that = this;

markStudent(studentIds).then(response =>{

favoriteStudents.forEach(student => {

student.favorite = {key: '未收藏', value: false};

});

that.$message({

type:'success',

message:'已取消收藏'

})

})

},

handleModifyStatus(row, status) {

this.$message({

message: '操作成功',

type: 'success'

});

row.status = status;

},

handleCreate() {

this.resetTemp();

this.dialogStatus = 'create';

this.dialogFormVisible = true;

},

handleUpdate(row) {

this.temp = Object.assign({}, row);

this.dialogStatus = 'update';

this.dialogFormVisible = true;

},

// handleDelete(row) {

// let confirmMessage = '您将删除学号为\'' + row.studentId + '\' '

// + row.studentName + ' 的所有信息,是否继续?';

// let that =this;

// let feedbackMessage = '';

// this.$confirm(confirmMessage,'删除学生',{

// confirmButtonText:'确定',

// cancelButtonText:'取消',

// type:'warning',

// beforeClose: (action,instance,done) =>{

// if(action==='confirm'){

// instance.confirmButtonLoading = true;

// return new Promise((resolve, reject) => {

// let studentIds = [];

// studentIds.push(row.studentId);

// deleteStudentBatch(studentIds).then((response) => {

// instance.confirmButtonLoading = false;

// this.$message({

// message: '删除成功',

// type: 'success',

// duration: 1500

// });

// done();

// resolve(response);

// }).catch(error => {

// instance.confirmButtonLoading = false;

// done();

// this.$message({

// message: error,

// type: 'error',

// duration: 1500

// });

// })

// });

// }

// done();

// }

// }).then(() =>{

// const index = that.studentList.indexOf(row);

// that.studentList.splice(index, 1);

// }).catch(() =>{

// that.$message({

// type:'info',

// message:'取消删除'

// })

// });

// },

handleDelete(index) {

let row = this.studentList[index];

let confirmMessage = '您将删除学号为\'' + row.studentId + '\' '

+ row.studentName + ' 的所有信息,是否继续?';

let that =this;

let feedbackMessage = '';

this.$confirm(confirmMessage,'删除学生',{

confirmButtonText:'确定',

cancelButtonText:'取消',

type:'warning',

beforeClose: (action,instance,done) =>{

if(action==='confirm'){

instance.confirmButtonLoading = true;

return new Promise((resolve, reject) => {

let studentIds = [];

studentIds.push(row.studentId);

deleteStudentBatch(studentIds).then((response) => {

instance.confirmButtonLoading = false;

this.$message({

message: '删除成功',

type: 'success',

duration: 1500

});

done();

resolve(response);

}).catch(error => {

instance.confirmButtonLoading = false;

done();

this.$message({

message: error,

type: 'error',

duration: 1500

});

})

});

}

done();

}

}).then(() =>{

that.studentList.splice(index, 1);

}).catch(() =>{

that.$message({

type:'info',

message:'取消删除'

})

});

},

create() {

createStudent(this.temp).then(response => {

this.$message({

message:'添加成功',

type:'success',

duration:1500

});

// this.studentList.unshift(this.temp);

this.dialogFormVisible = false;

});

},

update() {

updateStudent(this.temp).then(response =>{

for (const v of this.studentList) {

if (v.studentId === this.temp.studentId) {

const index = this.studentList.indexOf(v);

// this.studentList.splice(index, 1, this.temp);

break;

}

}

this.$message({

type: 'success',

message:'更新成功'

});

});

this.dialogFormVisible = false;

},

resetTemp() {

this.temp = {

studentId:'',

studentName:'',

grade:'',

className:'',

profession:'',

status:{

statusId:3,

chineseValue:'空闲',

englishValue:'available',

},

favorite:{

favoriteId:0,

chineseValue:'已收藏',

englishValue:'favorite'

},

finishedTaskCount:0

};

},

handleFetchPv(pv) {

fetchPv(pv).then(response => {

this.pvData = response.data.pvData;

this.dialogPvVisible = true;

})

},

handleDownload() {

require.ensure([], () => {

const { export\_json\_to\_excel } = require('vendor/Export2Excel');

const tHeader = ['学号', '姓名', '年级', '专业', '班级','参与任务数'];

const filterVal = ['studentId', 'studentName', 'grade', 'profession', 'className','finishedTaskCount'];

const data = this.formatJson(filterVal, this.list);

export\_json\_to\_excel(tHeader, data, '学生基本信息表');

})

},

formatJson(filterVal, jsonData) {

return jsonData.map(v => filterVal.map(j => {

if (j === 'timestamp') {

return parseTime(v[j])

} else {

return v[j]

}

}))

},

}

}

</script>

<style lang="scss">

.student-list-container{

margin-top: 90px;

}

.btn-import-container {

padding-right: 20px;

float:right;

.btn-item {

display: inline-block;

vertical-align: middle;

margin-bottom: 10px;

margin-left: 5px;

}

}

.test{

padding: 20px;

}

.title{

margin-bottom: 20px;

width:22%;

font:bold 36px 微软雅黑;

}

div{

background-color:transparent;

margin:0px auto;

}

</style>

## 数据挖掘模块

<template>

<div>

<el-row>

<full-page ref="fullpage" :options="pageOptions">

<div class="section">

<el-row style="margin-top: 20px">

<el-col :offset="20" :span="3">

<el-select size="medium"

clearable

v-model="xName"

placeholder="排序">

<el-option v-for="item in xOptions" :key="item.key" :label="item.label" :value="item.key">

</el-option>

</el-select>

</el-col>

</el-row>

<el-row>

<el-col>

<scatter :height="'940px'" :option="\_histogramOption">

</scatter>

</el-col>

</el-row>

</div>

<div class="section">

<el-row style="margin-top: 20px">

<el-col :offset="20" :span="3">

<el-select size="medium"

clearable

v-model="xName"

placeholder="排序">

<el-option v-for="item in xOptions" :key="item.key" :label="item.label" :value="item.key">

</el-option>

</el-select>

</el-col>

</el-row>

<el-row>

<el-col>

<scatter :height="'960px'" :option="\_boxOption">

</scatter>

</el-col>

</el-row>

</div>

</full-page>

</el-row>

</div>

</template>

<script>

import Scatter from '@/components/Charts/scatter'

import EchartStat from 'echarts-stat'

import dataTool from 'echarts/dist/extension/dataTool'

import { Loading } from 'element-ui';

import echarts from 'echarts';

var CATEGORY\_DIM\_COUNT = 6;

var GAP = 1;

var BASE\_LEFT = 5;

var BASE\_TOP = 10;

// var GRID\_WIDTH = 220;

// var GRID\_HEIGHT = 220;

var GRID\_WIDTH = (100 - BASE\_LEFT - GAP) / CATEGORY\_DIM\_COUNT - GAP;

var GRID\_HEIGHT = (100 - BASE\_TOP - GAP) / CATEGORY\_DIM\_COUNT - GAP;

var CATEGORY\_DIM = 7;

var SYMBOL\_SIZE = 3;

export default {

name: "density",

components: {

Scatter

},

props: {

setName: {

type: String,

default: 'Linear Regression'

}

},

beforeRouteEnter (to, from, next) {

Loading.service({

lock: true,

text: '正在生成视图,请稍后...',

spinner: 'el-icon-loading',

background: 'rgba(0, 0, 0, 0.7)'

});

next(vm => {})

},

activated(){

this.$nextTick(() => { // 以服务的方式调用的 Loading 需要异步关闭

Loading.service({

lock: true,

text: '正在生成视图,请稍后...',

spinner: 'el-icon-loading',

background: 'rgba(0, 0, 0, 0.7)'

}).close();

});

},

mounted: function () {

this.$nextTick(() => { // 以服务的方式调用的 Loading 需要异步关闭

Loading.service({

lock: true,

text: '正在生成视图,请稍后...',

spinner: 'el-icon-loading',

background: 'rgba(0, 0, 0, 0.7)'

}).close();

});

},

data() {

return {

xOptions: [{

key: '1',

label: '1',

},

{

key: '2',

label: '2',

},

{

key: '3',

label: '3',

},

{

key: '4',

label: '4',

}],

yOptions: [{

key: '1',

label: '1',

},

{

key: '2',

label: '2',

},

{

key: '3',

label: '3',

},

{

key: '4',

label: '4',

}],

pageOptions: {

licenseKey: 'OPEN-SOURCE-GPLV3-LICENSE',

menu: true,

navigation: true,

},

xName: 'x属性',

yName: 'y属性',

girth: [8.3, 8.6, 8.8, 10.5, 10.7, 10.8, 11.0, 11.0, 11.1, 11.2, 11.3, 11.4, 11.4, 11.7, 12.0, 12.9, 12.9, 13.3, 13.7, 13.8, 14.0, 14.2, 14.5, 16.0, 16.3, 17.3, 17.5, 17.9, 18.0, 18.0, 20.6]

}

},

methods: {

formatter(param) {

return [

'Experiment ' + param.name + ': ',

'upper: ' + param.data[0],

'Q1: ' + param.data[1],

'median: ' + param.data[2],

'Q3: ' + param.data[3],

'lower: ' + param.data[4]

].join('<br/>')

}

},

computed: {

\_histogramReg() {

let regression = EchartStat.histogram(this.girth);

return regression;

},

\_histogramOption() {

return {

title: {

text: 'Girths of Black Cherry Trees',

left: 'center',

},

color: ['rgb(25, 183, 207)'],

grid: {

left: '3%',

right: '3%',

bottom: '3%',

containLabel: true

},

xAxis: [{

type: 'value',

scale: true, //这个一定要设，不然barWidth和bins对应不上

}],

yAxis: [{

type: 'value',

}],

series: [{

name: 'height',

type: 'bar',

barWidth: '99.3%',

label: {

normal: {

show: true,

position: 'insideTop',

formatter: function (params) {

return params.value[1];

}

}

},

data: this.\_histogramReg.data

}]

}

},

\_boxOption() {

return {

title: {

text: 'Multiple Categories',

left: 'center',

},

legend: {

y: '10%',

data: ['category0', 'category1', 'category2', 'category3']

},

tooltip: {

trigger: 'item',

axisPointer: {

type: 'shadow'

}

},

grid: {

left: '10%',

top: '20%',

right: '10%',

bottom: '15%'

},

xAxis: {

type: 'category',

data: this.\_boxData[0].axisData,

boundaryGap: true,

nameGap: 30,

splitArea: {

show: true

},

axisLabel: {

formatter: 'expr {value}'

},

splitLine: {

show: false

}

},

yAxis: {

type: 'value',

name: 'Value',

min: -400,

max: 600,

splitArea: {

show: false

}

},

dataZoom: [

{

type: 'inside',

start: 0,

end: 20

},

{

show: true,

height: 20,

type: 'slider',

top: '90%',

xAxisIndex: [0],

start: 0,

end: 20

}

],

series: [

{

name: 'category0',

type: 'boxplot',

data: this.\_boxData[0].boxData,

tooltip: {formatter: this.formatter}

},

{

name: 'category1',

type: 'boxplot',

data: this.\_boxData[1].boxData,

tooltip: {formatter: this.formatter}

},

{

name: 'category2',

type: 'boxplot',

data: this.\_boxData[2].boxData,

tooltip: {formatter: this.formatter}

},

{

name: 'outlier',

type: 'pictorialBar',

symbolPosition: 'end',

symbolSize: 8,

data: this.\_boxData[0].outliers

},

{

name: 'outlier',

type: 'pictorialBar',

symbolPosition: 'end',

symbolSize: 8,

data: this.\_boxData[1].outliers

},

{

name: 'outlier',

type: 'pictorialBar',

symbolPosition: 'end',

symbolSize: 8,

barGap: '30%',

data: this.\_boxData[2].outliers

}

]

}

},

\_boxData() {

let data = [];

for (let seriesIndex = 0; seriesIndex < 5; seriesIndex++) {

let seriesData = [];

for (let i = 0; i < 18; i++) {

let cate = [];

for (let j = 0; j < 90; j++) {

cate.push(Math.random() \* 200);

}

for (let j = 0; j < 10; j++) {

cate.push(Math.random() \* 400);

}

seriesData.push(cate);

}

data.push(dataTool.prepareBoxplotData(seriesData));

}

return data;

}

}

}

</script>

<style scoped>

</style>

## 分组模块

<template>

<div class="calendar-list-container test">

<group-modal :to-group="\_updatingTargetGroup"

@on-closed="handleClosed"

@on-confirm="handleUpdateConfirm"

:to-show="showEditModal"

:to-candidates="allStudents"

:to-group-members="\_updatingTargetGroup.groupMembers"

:to-task-status="taskStatusOptions">

</group-modal>

<el-row>

<el-col :offset="9" :span="14">

<el-row>

<el-col>

<el-row :gutter="20">

<el-col :span="9">

<el-autocomplete

size="medium"

style="width: 100%"

v-model="listQuery.groupName"

suffix-icon="el-icon-edit"

:fetch-suggestions="groupNameSearch"

select-when-unmatched

:placeholder="$t('p.group.list.filter.groupName')">

</el-autocomplete>

</el-col>

<el-col :span="9">

<el-autocomplete

size="medium"

style="width: 100%"

suffix-icon="el-icon-zoom-in"

v-model="wrappedSuggestedGroupLeader"

:fetch-suggestions="groupLeaderSearch"

@select="handleLeaderFilter"

select-when-unmatched

:placeholder="$t('p.group.list.filter.leader')">

</el-autocomplete>

</el-col>

<el-col :span="6">

<el-select size="medium"

clearable

v-model="listQuery.sort"

placeholder="排序">

<el-option v-for="item in sortOptions" :key="item.key" :label="item.label" :value="item.key">

</el-option>

</el-select>

</el-col>

<el-col>

</el-col>

</el-row>

<el-row :gutter="20" style="margin-top: 20px">

<el-col :span="6">

<el-select

size="medium"

clearable

v-model="listQuery.taskStatus">

<el-option

v-for="item in taskStatusOptions"

:value="item.value"

:key="item.value"

:label="item.description">

</el-option>

</el-select>

</el-col>

<el-col :span="14">

<el-date-picker

clearable

size="medium"

style="width: 100%"

v-model="value7"

@change="handTimeFilter"

type="daterange"

unlink-panels

format="yyyy 年 MM 月 dd 日 HH:mm:ss"

value-format="yyyy-MM-dd HH:mm:ss"

range-separator="至"

:start-placeholder="$t('p.group.list.filter.beginDate')"

:end-placeholder="$t('p.group.list.filter.endDate')"

:picker-options="pickerOptions">

</el-date-picker>

</el-col>

<el-col :span="1">

<Button style="margin-top: 2px" type="primary" shape="circle" icon="ios-search"

@click="handleFilter"></Button>

</el-col>

<el-col :span="1" style="margin-left: 10px">

<el-button size="medium"

class="btn-item" type="danger"

plain

round

icon="el-icon-circle-close-outline"

@click="handleBatchDelete"

:disabled="!\_selectionIds.length">批量删除

</el-button>

</el-col>

</el-row>

</el-col>

</el-row>

</el-col>

</el-row>

<el-row style="margin-top: 20px">

<Table border

:loading="listLoading"

:columns="groupColumns"

:data="groupList"

size='large'

@on-selection-change="handleSelectionChange"

stripe></Table>

</el-row>

<el-row style="float:right;margin-top: 20px">

<el-col>

<el-pagination @size-change="handleSizeChange"

background

size='large'

@current-change="handleCurrentChange"

:page-sizes="[10,20,30, 50]"

:page-size="listQuery.size"

layout="total, sizes, prev, pager, next, jumper"

:total="total">

</el-pagination>

</el-col>

</el-row>

<el-row id="group-view" style="margin-top: 60px">

<el-col>

<transition

mode="out-in"

name="custom-classes-transition"

enter-active-class="animated bounceIn"

leave-active-class="animated bounceOutRight">

<teacher-group-view

v-if="\_wrappedDetailTarget"

:to-stages="stages"

:task-id="\_wrappedDetailTarget[0].taskId"

:groups="\_wrappedDetailTarget"

:key="this.\_wrappedDetailTarget[0].groupId">

</teacher-group-view>

</transition>

</el-col>

</el-row>

</div>

</template>

<script>

import {parseTime, deleteEmptyProperty} from 'utils';

import {fetchStudentList} from 'api/students';

import {

getGroupList,

getLeisureStudents,

createGroupPreview,

getGroupPreview,

getGroup,

deleteGroups,

updateGroup,

getMembers,

updateLeader,

configureMembers,

getGroupNames,

getGroupLeaders,

getTaskStatus

} from 'api/groups';

import {

getRefCollections,

getRefStages,

getRefGroups, fetchConfiguredAlgortithms

} from 'api/tasks';

import TeacherGroupView from '../components/teacherGroupView';

import GroupModal from '../components/groupModal';

// import TeacherGroupView from "../components/teacherGroupView.vue";

export default {

name: 'group-list',

components: {

TeacherGroupView,

GroupModal},

data() {

let vm = this;

return {

allStudents: [],

groupModel:{

groupName:'',

groupLeader:{

studentId:'',

studentName:'',

className:'',

},

taskStatus:'',

groupMembers:[],

},

pickerOptions: {

shortcuts: [{

text: '最近一周',

onClick(picker) {

const end = new Date();

const start = new Date();