<template>

    <div class="login">

        <div class="main">

            <div class="left">

                <h1>知识产权从业人员信息化管理系统</h1>

            </div>

            <div class="right">

                <el-form ref="ruleFormRef" :model="ruleForm" :rules="rules" label-width="80px" class="demo-ruleForm">

                    <el-form-item label="用户名" prop="username" style="margin-top: 70px;">

                        <el-input v-model="ruleForm.username" />

                    </el-form-item>

                    <el-form-item label="密码" prop="password" style="margin-top: 30px;">

                        <el-input v-model="ruleForm.password" type="password" />

                    </el-form-item>

                    <el-button @click="handleLogin" type="primary"

                        style="margin-top: 30px;width: 380px;margin-left: 80px;">

                        登录</el-button>

                </el-form>

            </div>

        </div>

    </div>

    <Particles id="tsparticles" class="login-partic" :options="options" />

</template>

<script lang="ts">

export default {

    name: "Login"

}

</script>

<script lang="ts" setup>

import { ref, reactive } from 'vue'

import { useRouter } from 'vue-router'

import { ElMessage, FormInstance, FormRules } from 'element-plus'

import axios from '../http'

const options = reactive({})

const router = useRouter()

const ruleForm = reactive({

    username: '',

    password: ''

})

const rules = reactive<FormRules>({

    username: [

        {

            trigger: 'blur',

            required: true,

            validator: (rule: any, value: any, callback: Function) => {

                var passwordreg = /^\w+$/

                if (!passwordreg.test(value)) {

                    callback(

                        new Error(

                            '用户名必须英文字母、数字、下划线组成!'

                        )

                    )

                } else {

                    callback()

                }

            }

        },

        { min: 5, max: 8, message: '请输入5-8位用户名', trigger: 'blur' },

    ],

    password: [

        {

            trigger: 'blur',

            required: true,

            validator: (rule: any, value: any, callback: Function) => {

                var passwordreg = /^\w+$/

                if (!passwordreg.test(value)) {

                    callback(

                        new Error(

                            '用户名必须英文字母、数字、下划线组成!'

                        )

                    )

                } else {

                    callback()

                }

            }

        },

        { min: 6, max: 10, message: '请输入6-10位用户密码', trigger: 'blur' },

    ],

})

const ruleFormRef = ref(null)

const handleLogin = () => {

    (ruleFormRef.value as any).validate((valid: any, fields: any) => {

        if (valid) {

            axios.post('/login.json', { ...ruleForm }).then((res: any) => {

                ElMessage({

                    type: res.data.status === 1 ? 'success' : 'error',

                    message: res.data.message

                })

                if (res.data.status === 1) {

                    router.push({ name: 'home' })

                    sessionStorage.setItem('userinfo', JSON.stringify(ruleForm))

                }

            })

        }

    })

}

</script>

<style lang="less" scoped>

.login {

    height: 100%;

    background-image: url(@/assets/img/bg.png);

    background-repeat: no-repeat;

    background-size: 100% 100%;

    overflow: hidden;

    .main {

        margin-top: 150px;

        display: flex;

        justify-content: space-around;

        align-items: center;

        .right {

            width: 500px;

            height: 350px;

            background: #fff;

            border-radius: 10px;

            padding-right: 40px;

            box-shadow: 2px 2px 10px #06C;

        }

    }

    .footer {

        margin: 0 auto;

        margin-top: 100px;

        text-align: center;

    }

}

</style>

<template>

    <div class="charts">

        <div class="container " ref="container1"></div>

        <div class="container " ref="container2"></div>

    </div>

</template>

<script lang="ts">

export default {

    name: ""

}

</script>

<script lang="ts" setup>

import { ref, reactive, onMounted } from 'vue'

import \* as echarts from 'echarts';

import axios from '../../http';

const container1 = ref<HTMLElement | null>(null)

const container2 = ref<HTMLElement | null>(null)

const initChart = () => {

    axios.get('/charts.json', { params: {} }).then(res => {

        const myChart1 = echarts.init(container1.value as HTMLElement)

        myChart1.setOption(res.data.chart1);

        const myChart2 = echarts.init(container2.value as HTMLElement)

        myChart2.setOption(res.data.chart2)

    })

}

onMounted(() => {

    initChart()

})

</script>

<style lang="less" scoped>

.charts {

    width: 100%;

    display: flex;

    align-items: center;

    justify-content: space-around;

}

.container {

    width: 400px;

    height: 400px;

}

</style>

<template>

    <div class="info">

        <el-breadcrumb :separator-icon="ArrowRight">

            <el-breadcrumb-item :to="{ path: '/' }">主页</el-breadcrumb-item>

            <el-breadcrumb-item>基本信息</el-breadcrumb-item>

        </el-breadcrumb>

        <div class="form-wrap">

            <div class="btn-group">

                <el-button @click="add" type="success" style="width: 100px;">添加信息</el-button>

                <el-button style="width: 100px;">导入</el-button>

                <el-button style="width: 100px;">导出</el-button>

            </div>

            <el-table :data="tableData" style="width: 100%" :row-class-name="tableRowClassName">

                <el-table-column prop="id" label="编号" width="55px" />

                <el-table-column prop="name" label="姓名" width="80px" />

                <el-table-column prop="gender" label="性别" width="80px">

                    <template #default="{ row, column, $index }">

                        <span>

                            {{row.gender===1?'男':'女'}}

                        </span>

                    </template>

                </el-table-column>

                <el-table-column prop="educational" label="学历" width="120px">

                    <template #default="{ row, column, $index }">

                        <span>

                            {{row.educational===0?'初中及以下':row.educational===1?'高中':row.educational===2?'统招专科':row.educational===3?'统招本科':row.educational===4?'硕士及以上':"其他"}}

                        </span>

                    </template>

                </el-table-column>

                <el-table-column prop="unit" label="单位名称" />

                <el-table-column prop="phone" label="联系电话" />

                <el-table-column label="操作" width="120px">

                    <template #default="{ row, column, $index }">

                        <el-button link type="primary" size="small" @click="edit(row)">编辑</el-button>

                        <el-button link type="primary" size="small" @click="del(row)">删除</el-button>

                    </template>

                </el-table-column>

            </el-table>

            <el-pagination v-model:currentPage="currentPage" v-model:page-size="pageSize" :page-sizes="[7, 14, 21, 28]"

                background layout="total, sizes, prev, pager, next, jumper" :total="totalPage"

                @size-change="handleSizeChange" @current-change="handleCurrentChange" />

            <el-dialog v-model="dialogshow" width="770px">

                <el-form :model="form" ref="formRef" label-width="100px" :rules="rules">

                    <div class="form">

                        <div class="company">

                            <div class="title">企业信息</div>

                            <el-form-item label="单位名称" prop="unit">

                                <el-input v-model="form.unit" placeholder="请填写单位名称" />

                            </el-form-item>

                            <el-form-item label="单位地址" prop="address">

                                <el-input v-model="form.address" placeholder="请填写单位地址" />

                            </el-form-item>

                            <el-form-item label="联系人" prop="linkman">

                                <el-input v-model="form.linkman" placeholder="请填写联系人" />

                            </el-form-item>

                            <el-form-item label="联系电话" prop="phone">

                                <el-input v-model="form.phone" placeholder="请填写联系电话" />

                            </el-form-item>

                        </div>

                        <div class="personal">

                            <div class="title">个人信息</div>

                            <el-form-item label="姓名" prop="name">

                                <el-input v-model="form.name" placeholder="请填写人员姓名" />

                            </el-form-item>

                            <el-form-item label="性别" prop="gender">

                                <el-radio-group v-model="form.gender" placeholder="请选择人员性别">

                                    <el-radio :label="1">男</el-radio>

                                    <el-radio :label="0">女</el-radio>

                                </el-radio-group>

                            </el-form-item>

                            <el-form-item label="学历" prop="educational" style="width:100%">

                                <el-select v-model="form.educational" placeholder="请选择人员学历">

                                    <el-option v-for="(item,index) in educational" :label="item.message"

                                        :value="item.value" />

                                </el-select>

                            </el-form-item>

                            <el-form-item label="专业" prop="major">

                                <el-input v-model="form.major" placeholder="请填写人员专业" />

                            </el-form-item>

                            <el-form-item label="个人简介" prop="remark">

                                <el-input v-model="form.remark" type="textarea" />

                            </el-form-item>

                        </div>

                    </div>

                </el-form>

                <template #footer>

                    <span class="dialog-footer">

                        <el-button type="primary" @click="onSubmit" style="width:100px">确认</el-button>

                        <el-button @click="onCancel" style="width:100px">取消</el-button>

                    </span>

                </template>

            </el-dialog>

        </div>

    </div>

</template>

<script lang="ts">

export default {

    name: ""

}

</script>

<script setup lang="ts">

import { ref, reactive, nextTick } from 'vue'

import { ArrowRight } from '@element-plus/icons-vue'

import { ElMessage, FormRules, ElMessageBox } from 'element-plus'

import axios from '../../http'

const formRef = ref(null)

const tableData = ref([])

const currentPage = ref(1)

const totalPage = ref(1)

const pageSize = ref(7)

const handleSizeChange = (evt: any) => {

    pageSize.value = evt

    getInfo()

}

const handleCurrentChange = (evt: any) => {

    currentPage.value = evt

    getInfo()

}

const getInfo = () => {

    axios.get('/getinfo.json', { params: { page: currentPage.value, pageSize: pageSize.value } }).then(res => {

        totalPage.value = res.data.total

        tableData.value = res.data.result

    })

}

getInfo()

const dialogshow = ref(false)

const isEdit = ref(false)

const add = () => {

    dialogshow.value = true;

    form.value = JSON.parse(JSON.stringify(defaultForm))

    isEdit.value = false

}

const edit = (row: any) => {

    dialogshow.value = true;

    form.value = JSON.parse(JSON.stringify(row));

    isEdit.value = true

    // setTimeout(() => {

    //     (formRef.value as any).resetFields()

    // }, 100)

}

const del = (row: any) => {

    ElMessageBox.confirm(`确定删除【${row.name}】吗？`, {

        confirmButtonText: '确认',

        cancelButtonText: '取消',

    }).then(() => {

        axios.get('/deleteinfo.json', { params: { id: row.id } }).then((res: any) => {

            getInfo()

            ElMessage({

                type: 'success',

                message: res.data.message,

            })

        })

    })

}

const defaultForm = {

    name: '',

    educational: 3,

    major: '',

    unit: '',

    remark: '',

    address: '',

    linkman: '',

    phone: '',

    gender: 1

}

const form = ref({

    name: '',

    educational: '',

    major: '',

    unit: '',

    remark: '',

    address: '',

    linkman: '',

    phone: '',

    gender: ""

})

const onSubmit = () => {

    (formRef.value as any).validate((valid: any, fields: any) => {

        if (valid) {

            let url

            if (isEdit.value) {

                url = '/editinfo.json'

            } else {

                url = '/addinfo.json'

            }

            axios.post(url, {

                ...JSON.parse(JSON.stringify(form.value))

            }).then((res: any) => {

                getInfo()

                ElMessage({

                    type: 'success',

                    message: res.data.message

                })

                form.value = JSON.parse(JSON.stringify(defaultForm))

                setTimeout(() => {

                    (formRef.value as any).resetFields()

                    dialogshow.value = false

                }, 100)

            })

        }

    })

}

const onCancel = () => {

    dialogshow.value = false

    setTimeout(() => {

        form.value = JSON.parse(JSON.stringify(defaultForm))

    }, 100)

}

const educational = reactive([

    { value: 0, message: '初中及以下' },

    { value: 1, message: '高中' },

    { value: 2, message: '统招专科' },

    { value: 3, message: '统招本科' },

    { value: 4, message: '硕士及以上' },

    { value: 5, message: '其他' },

])

const rules = reactive<FormRules>({

    name: [{ required: true, message: '请填写人员姓名', trigger: 'blur' }],

    educational: [{ required: true, message: '请填写人员姓名', trigger: 'change' }],

    major: [{ required: true, message: '请填写人员专业', trigger: 'blur' }],

    unit: [{ required: true, message: '请填写单位名称', trigger: 'blur' }],

    address: [{ required: true, message: '请填写单位地址', trigger: 'blur' }],

    linkman: [{ required: true, message: '请填写联系人', trigger: 'blur' }],

    phone: [{ required: true, message: '请填写联系电话', trigger: 'blur' }],

    gender: [{ required: true, message: '请选择人员性别', trigger: 'change' }],

})

const tableRowClassName = ({

    row,

    rowIndex,

}: {

    row: any

    rowIndex: any

}) => {

    if ((rowIndex + 1) % 2 === 0) {

        // return 'warning-row'

        return 'success-row'

    }

    // else if (rowIndex % 3 === 0) {

    //     return 'success-row'

    // }

    return ''

}

</script>

<style lang="less" scoped>

.info {

    width: 100%;

    height: 100%;

    // overflow-y: auto;

    display: flex;

    flex-direction: column;

    .form-wrap {

        display: flex;

        flex-direction: column;

        justify-content: space-around;

        padding: 0 50px;

        .btn-group {

            margin-bottom: 20px;

        }

        .el-table {

            height: 360px;

            overflow-y: auto;

        }

        .el-pagination {

            align-self: flex-end;

            padding: 10px 0;

        }

        .el-form {

            padding: 20px;

            width: 750px;

            .form {

                display: flex;

                .title {

                    text-align: center;

                    font-weight: bold;

                }

                >div {

                    flex: 1;

                }

            }

            .el-form-item {

                margin-top: 30px;

                .el-select {

                    width: 100%;

                }

            }

        }

    }

}

</style>

<template>

    <div class="record">

        <el-breadcrumb :separator-icon="ArrowRight">

            <el-breadcrumb-item :to="{ path: '/' }">主页</el-breadcrumb-item>

            <el-breadcrumb-item>工作履历</el-breadcrumb-item>

        </el-breadcrumb>

        <div class="form-wrap">

            <div class="btn-group">

                <el-button @click="add" type="success" style="width: 100px;">添加经历</el-button>

                <el-button style="width: 100px;">导入</el-button>

                <el-button style="width: 100px;">导出</el-button>

            </div>

            <el-table :data="tableData" style="width: 100%" :row-class-name="tableRowClassName">

                <el-table-column prop="id" label="编号" width="55px" />

                <el-table-column prop="userid" label="对应人员" width="80px">

                    <template #default="{ row, column, $index }">

                        <span v-if="userData.find((item:any)=>item.id===row.userid)">

                            {{(userData.find((item:any)=>item.id===row.userid) as any).name}}

                        </span>

                    </template>

                </el-table-column>

                <el-table-column prop="time" label="工作时间">

                    <template #default="{ row, column, $index }">

                        <span>

                            {{row.time[0] +'至'+row.time[1]}}

                        </span>

                    </template>

                </el-table-column>

                <el-table-column prop="company" label="公司名称" />

                <el-table-column prop="position" label="职位名称" />

                <el-table-column label="操作" width="120px">

                    <template #default="{ row, column, $index }">

                        <el-button link type="primary" size="small" @click="edit(row)">编辑</el-button>

                        <el-button link type="primary" size="small" @click="del(row)">删除</el-button>

                    </template>

                </el-table-column>

            </el-table>

            <el-pagination v-model:currentPage="currentPage" v-model:page-size="pageSize" :page-sizes="[7, 14, 21, 28]"

                background layout="total, sizes, prev, pager, next, jumper" :total="totalPage"

                @size-change="handleSizeChange" @current-change="handleCurrentChange" />

            <el-dialog v-model="dialogshow" width="580px">

                <el-form ref="formRef" :model="form" label-width="100px">

                    <el-form-item :label="'对应人员'" prop="userid" :rules="{

                      required: true,

                      message: '请填写对应人员',

                      trigger: 'blur',

                    }">

                        <el-select v-model="form.userid" placeholder="请选择对应人员" :disabled="isEdit">

                            <el-option v-for="(item,index) in userData" :label="item.name" :value="item.id" />

                        </el-select>

                    </el-form-item>

                    <el-form-item :label="'工作时间'" prop="time" :rules="{

                      required: true,

                      message: '请填写工作时间',

                      trigger: 'blur',

                    }">

                        <el-date-picker v-model="form.time" format="YYYY/MM/DD" value-format="YYYY-MM-DD"

                            type="daterange" start-placeholder="开始日期" end-placeholder="结束日期"

                            :default-value="[new Date(2020, 9, 1), new Date(2022, 10, 1)]" />

                    </el-form-item>

                    <!-- <el-form-item :label="'公司名称'" :prop="'record.'+index+'.company'" :rules="{ -->

                    <el-form-item :label="'公司名称'" prop="company" :rules="{

                      required: true,

                      message: '请填写公司名称',

                      trigger: 'blur',

                    }">

                        <el-input v-model="form.company" placeholder="请填写公司名称" />

                    </el-form-item>

                    <el-form-item :label="'职位名称'" prop="position" :rules="{

                      required: true,

                      message: '请填写职位名称',

                      trigger: 'blur',

                    }">

                        <el-input v-model="form.position" placeholder="请填写职位名称" />

                    </el-form-item>

                </el-form>

                <template #footer>

                    <span class="dialog-footer">

                        <el-button type="primary" @click="onSubmit" style="width:100px">确认</el-button>

                        <el-button @click="onCancel" style="width:100px">取消</el-button>

                    </span>

                </template>

            </el-dialog>

        </div>

    </div>

</template>

<script lang="ts">

export default {

    name: ""

}

</script>

<script setup lang="ts">

import { ref, reactive } from 'vue'

import { ArrowRight, Delete } from '@element-plus/icons-vue'

import { ElMessage, ElMessageBox } from 'element-plus'

import axios from '../../http'

interface IPubilc {

    [k: string]: any

}

const userData = ref<IPubilc[]>([])

axios.get('/getinfo.json', { params: { page: 1, pageSize: 10000 } }).then(res => {

    userData.value = res.data.result

})

const currentPage = ref(1)

const pageSize = ref(7)

const totalPage = ref(1)

const handleSizeChange = (evt: any) => {

    pageSize.value = evt

    getData()

}

const handleCurrentChange = (evt: any) => {

    currentPage.value = evt

    getData()

}

const tableData = ref([])

const getData = () => {

    axios.get('/getrecord.json', { params: { page: currentPage.value, pageSize: pageSize.value } }).then(res => {

        tableData.value = res.data.result

        totalPage.value = res.data.total

    })

}

getData()

const dialogshow = ref(false)

const edit = (row: any) => {

    dialogshow.value = true;

    form.value = JSON.parse(JSON.stringify(row))

    isEdit.value = true

}

const del = (row: any) => {

    ElMessageBox.confirm(`确定删除【当前经历】吗？`, {

        confirmButtonText: '确认',

        cancelButtonText: '取消',

    }).then(() => {

        axios.get('/deleterecord.json', { params: { id: row.id, userid: row.userid } }).then((res: any) => {

            getData()

            ElMessage({

                type: 'success',

                message: res.data.message,

            })

        })

    })

}

const add = () => {

    dialogshow.value = true;

    form.value = JSON.parse(JSON.stringify(defaultForm))

    isEdit.value = false

}

const defaultForm = {

    time: '',

    company: '',

    position: '',

    userid: 2

}

const form = ref({

    time: '',

    company: '',

    position: '',

    userid: 2

})

const isEdit = ref(false)

const onSubmit = () => {

    (formRef.value as any).validate((valid: any, fields: any) => {

        if (valid) {

            let url

            if (isEdit.value) {

                url = '/editrecord.json'

            } else {

                url = '/addrecord.json'

            }

            axios.post(url, {

                ...JSON.parse(JSON.stringify(form.value))

            }).then((res: any) => {

                getData()

                ElMessage({

                    type: 'success',

                    message: res.data.message

                })

                form.value = JSON.parse(JSON.stringify(defaultForm))

                dialogshow.value = false

                setTimeout(() => {

                    (formRef.value as any).resetFields()

                }, 100)

            })

        }

    })

}

const formRef = ref(null)

const onCancel = () => {

    dialogshow.value = false

    setTimeout(() => {

        form.value = JSON.parse(JSON.stringify(defaultForm))

    }, 100)

}

const tableRowClassName = ({

    row,

    rowIndex,

}: {

    row: any

    rowIndex: any

}) => {

    if ((rowIndex + 1) % 2 === 0) {        // return 'warning-row'

        return 'success-row'

    }

    // else if (rowIndex % 3 === 0) {

    //     return 'success-row'

    // }

    return ''

}

</script>

<style lang="less" scoped>

.record {

    height: 100%;

    width: 100%;

    display: flex;

    flex-direction: column;

    overflow-y: auto;

    .form-wrap {

        width: 100%;

        display: flex;

        flex-direction: column;

        justify-content: space-around;

        padding: 0 50px;

        .btn-group {

            margin-bottom: 20px;

        }

        .el-table {

            height: 360px;

            overflow-y: auto;

        }

        .el-pagination {

            align-self: flex-end;

            padding: 10px 0;

        }

        .el-form {

            width: 550px;

            .el-form-item {

                margin-bottom: 30px;

            }

            .record-item {

                box-shadow: 2px 2px 10px #06C;

                border: 1px solid #ccc;

                padding-right: 20px;

                border-radius: 10px;

                margin-bottom: 20px;

                display: flex;

                .title {

                    font-weight: bold;

                    display: flex;

                    align-items: center;

                    justify-content: space-between;

                    padding-left: 120px;

                    position: relative;

                    padding-top: 10px;

                    .mt-2 {

                        position: absolute;

                        right: -20px;

                        top: 0;

                        border-radius: 10px;

                    }

                }

                .box {

                    width: 100%;

                }

            }

        }

    }

}

</style>

<template>

    <div class="train">

        <el-breadcrumb :separator-icon="ArrowRight">

            <el-breadcrumb-item :to="{ path: '/' }">主页</el-breadcrumb-item>

            <el-breadcrumb-item>继续教育</el-breadcrumb-item>

        </el-breadcrumb>

        <div class="form-wrap">

            <div class="btn-group">

                <el-button @click="add" type="success" style="width: 100px;">添加经历</el-button>

                <el-button style="width: 100px;">导入</el-button>

                <el-button style="width: 100px;">导出</el-button>

            </div>

            <el-table :data="tableData" style="width: 100%" :row-class-name="tableRowClassName">

                <el-table-column prop="id" label="编号" width="55px" />

                <el-table-column prop="userid" label="对应人员" width="80px">

                    <template #default="{ row, column, $index }">

                        <span v-if="userData.find((item: any) => item.id === row.userid)">

                            {{ (userData.find((item: any) => item.id === row.userid) as any).name }}

                        </span>

                    </template>

                </el-table-column>

                <el-table-column prop="time" label="培训时间">

                    <template #default="{ row, column, $index }">

                        <span>

                            {{ row.time[0] + '至' + row.time[1] }}

                        </span>

                    </template>

                </el-table-column>

                <el-table-column prop="company" label="机构名称" />

                <el-table-column prop="position" label="培训内容" />

                <el-table-column label="操作" width="120px">

                    <template #default="{ row, column, $index }">

                        <el-button link type="primary" size="small" @click="edit(row)">编辑</el-button>

                        <el-button link type="primary" size="small" @click="del(row)">删除</el-button>

                    </template>

                </el-table-column>

            </el-table>

            <el-pagination v-model:currentPage="currentPage" v-model:page-size="pageSize" :page-sizes="[7, 14, 21, 28]"

                background layout="total, sizes, prev, pager, next, jumper" :total="totalPage"

                @size-change="handleSizeChange" @current-change="handleCurrentChange" />

            <el-dialog v-model="dialogshow" width="580px">

                <el-form ref="formRef" :model="form" label-width="100px">

                    <el-form-item :label="'对应人员'" prop="userid" :rules="{

                        required: true,

                        message: '请填写对应人员',

                        trigger: 'blur',

                    }">

                        <el-select v-model="form.userid" :disabled="isEdit" placeholder="请选择对应人员">

                            <el-option v-for="(item, index) in userData" :label="item.name" :value="item.id" />

                        </el-select>

                    </el-form-item>

                    <el-form-item :label="'培训时间'" prop="time" :rules="{

                        required: true,

                        message: '请填写培训时间',

                        trigger: 'blur',

                    }">

                        <el-date-picker v-model="form.time" format="YYYY/MM/DD" value-format="YYYY-MM-DD"

                            type="daterange" start-placeholder="开始日期" end-placeholder="结束日期"

                            :default-value="[new Date(2020, 9, 1), new Date(2022, 10, 1)]" />

                    </el-form-item>

                    <el-form-item :label="'机构名称'" prop="company" :rules="{

                        required: true,

                        message: '请填写机构名称',

                        trigger: 'blur',

                    }">

                        <el-input v-model="form.company" placeholder="请填写机构名称" />

                    </el-form-item>

                    <el-form-item :label="'培训内容'" prop="position" :rules="{

                        required: true,

                        message: '请填写培训内容',

                        trigger: 'blur',

                    }">

                        <el-input v-model="form.position" placeholder="请填写培训内容" />

                    </el-form-item>

                </el-form>

                <template #footer>

                    <span class="dialog-footer">

                        <el-button type="primary" @click="onSubmit" style="width:100px">确认</el-button>

                        <el-button @click="onCancel" style="width:100px">取消</el-button>

                    </span>

                </template>

            </el-dialog>

        </div>

    </div>

</template>

<script lang="ts">

export default {

    name: ""

}

</script>

<script setup lang="ts">

import { ref, reactive } from 'vue'

import { ArrowRight, Delete } from '@element-plus/icons-vue'

import { ElMessage, ElMessageBox } from 'element-plus'

import axios from '../../http'

interface IPubilc {

    [k: string]: any

}

const userData = ref<IPubilc[]>([])

axios.get('/getinfo.json', { params: { page: 1, pageSize: 10000 } }).then(res => {

    userData.value = res.data.result

})

const currentPage = ref(1)

const totalPage = ref(1)

const pageSize = ref(7)

const handleSizeChange = (evt: any) => {

    pageSize.value = evt

    getData()

}

const handleCurrentChange = (evt: any) => {

    currentPage.value = evt

    getData()

}

const tableData = ref([])

const getData = () => {

    axios.get('/gettrain.json', { params: { page: currentPage.value, pageSize: pageSize.value } }).then(res => {

        tableData.value = res.data.result

        totalPage.value = res.data.total

    })

}

getData()

const dialogshow = ref(false)

const isEdit = ref(false)

const edit = (row: any) => {

    dialogshow.value = true;

    form.value = JSON.parse(JSON.stringify(row))

    isEdit.value = true

}

const del = (row: any) => {

    ElMessageBox.confirm(`确定删除【当前经历】吗？`, {

        confirmButtonText: '确认',

        cancelButtonText: '取消',

    }).then(() => {

        axios.get('/deletetrain.json', { params: { id: row.id, userid: row.userid } }).then((res: any) => {

            getData()

            ElMessage({

                type: 'success',

                message: res.data.message,

            })

        })

    })

}

const add = () => {

    dialogshow.value = true;

    form.value = JSON.parse(JSON.stringify(defaultForm))

    isEdit.value = false

}

const defaultForm = {

    userid: 2,

    time: '',

    company: '',

    position: ''

}

const form = ref({

    userid: 2,

    time: '',

    company: '',

    position: ''

})

const formRef = ref(null)

const onCancel = () => {

    dialogshow.value = false

    setTimeout(() => {

        form.value = JSON.parse(JSON.stringify(defaultForm))

    }, 100)

}

const onSubmit = () => {

    (formRef.value as any).validate((valid: any, fields: any) => {

        if (valid) {

            let url

            if (isEdit.value) {

                url = '/edittrain.json'

            } else {

                url = '/addtrain.json'

            }

            axios.post(url, {

                ...JSON.parse(JSON.stringify(form.value))

            }).then((res: any) => {

                getData()

                ElMessage({

                    type: 'success',

                    message: res.data.message

                })

                form.value = JSON.parse(JSON.stringify(defaultForm))

                dialogshow.value = false

                setTimeout(() => {

                    (formRef.value as any).resetFields()

                }, 100)

            })

        }

    })

}

const tableRowClassName = ({

    row,

    rowIndex,

}: {

    row: any

    rowIndex: any

}) => {

    if ((rowIndex + 1) % 2 === 0) {

        // return 'warning-row'

        return 'success-row'

    }

    // else if (rowIndex % 3 === 0) {

    //     return 'success-row'

    // }

    return ''

}

</script>

<style lang="less" scoped>

.train {

    height: 100%;

    width: 100%;

    display: flex;

    flex-direction: column;

    overflow-y: auto;

    .form-wrap {

        width: 100%;

        display: flex;

        flex-direction: column;

        justify-content: space-around;

        padding: 0 50px;

        .btn-group {

            margin-bottom: 20px;

        }

        .el-table {

            height: 360px;

            overflow-y: auto;

        }

        .el-pagination {

            align-self: flex-end;

            padding: 10px 0;

        }

        .el-form {

            width: 550px;

            .el-form-item {

                margin-bottom: 30px;

            }

            .record-item {

                box-shadow: 2px 2px 10px #06C;

                border: 1px solid #ccc;

                padding-right: 20px;

                border-radius: 10px;

                margin-bottom: 20px;

                display: flex;

                .title {

                    font-weight: bold;

                    display: flex;

                    align-items: center;

                    justify-content: space-between;

                    padding-left: 120px;

                    position: relative;

                    padding-top: 10px;

                    .mt-2 {

                        position: absolute;

                        right: -20px;

                        top: 0;

                        border-radius: 10px;

                    }

                }

                .box {

                    width: 100%;

                }

            }

        }

    }

}

</style>

<template>

    <div class="case">

        <el-breadcrumb :separator-icon="ArrowRight">

            <el-breadcrumb-item :to="{ path: '/' }">主页</el-breadcrumb-item>

            <el-breadcrumb-item>经典案例</el-breadcrumb-item>

        </el-breadcrumb>

        <div class="form-wrap">

            <div class="btn-group">

                <el-button @click="add" type="success" style="width: 100px;">添加案例</el-button>

                <el-button style="width: 100px;">导入</el-button>

                <el-button style="width: 100px;">导出</el-button>

            </div>

            <el-table :data="tableData" style="width: 100%" :row-class-name="tableRowClassName">

                <el-table-column prop="id" label="编号" width="55px" />

                <el-table-column prop="userid" label="对应人员" width="80px">

                    <template #default="{ row, column, $index }">

                        <span v-if="userData.find((item: any) => item.id === row.userid)">

                            {{ (userData.find((item: any) => item.id === row.userid) as any).name }}

                        </span>

                    </template>

                </el-table-column>

                <el-table-column prop="name" label="案例名称" />

                <el-table-column prop="year" label="案例年份">

                    <template #default="{ row, column, $index }">

                        <span>

                            {{ row.year }}

                        </span>

                    </template>

                </el-table-column>

                <el-table-column prop="type" label="业务类型">

                    <template #default="{ row, column, $index }">

                        <span>

                            {{ row.type === 0 ? '专利' : row.type === 1 ? '商标' : '版权' }}

                        </span>

                    </template>

                </el-table-column>

                <el-table-column prop="intro" label="基本情况" />

                <el-table-column label="操作" width="120px">

                    <template #default="{ row, column, $index }">

                        <el-button link type="primary" size="small" @click="edit(row)">编辑</el-button>

                        <el-button link type="primary" size="small" @click="del(row)">删除</el-button>

                    </template>

                </el-table-column>

            </el-table>

            <el-pagination v-model:currentPage="currentPage" v-model:page-size="pageSize" :page-sizes="[7, 14, 21, 28]"

                background layout="total, sizes, prev, pager, next, jumper" :total="totalPage"

                @size-change="handleSizeChange" @current-change="handleCurrentChange" />

            <el-dialog v-model="dialogshow" width="580px">

                <el-form :model="form" ref="formRef" label-width="120px" :rules="rules">

                    <el-form-item :label="'对应人员'" prop="userid" :rules="{

                        required: true,

                        message: '请填写对应人员',

                        trigger: 'blur',

                    }">

                        <el-select v-model="form.userid" :disabled="isEdit" placeholder="请选择对应人员">

                            <el-option v-for="(item, index) in userData" :label="item.name" :value="item.id" />

                        </el-select>

                    </el-form-item>

                    <el-form-item label="案例名称" prop="name">

                        <el-input v-model="form.name" placeholder="请填写案例名称" />

                    </el-form-item>

                    <el-form-item class="date" label="案例年份" prop="year">

                        <el-date-picker v-model="form.year" type="year" format="YYYY" value-format="YYYY"

                            placeholder="请选择案例年份" />

                    </el-form-item>

                    <el-form-item label="所属业务类型" prop="type">

                        <el-select v-model="form.type" placeholder="请选择所属业务类型" style="width:100%">

                            <el-option v-for="(item, index) in type" :label="item.message" :value="item.value" />

                        </el-select>

                    </el-form-item>

                    <el-form-item label="案例基本情况" prop="intro">

                        <el-input v-model="form.intro" type="textarea" placeholder="请填写案例基本情况简介" />

                    </el-form-item>

                    <div class="bottom">

                        <el-form-item label="问题争议点" prop="error">

                            <el-input v-model="form.error" type="textarea" />

                        </el-form-item>

                        <el-form-item label="方案解决" prop="deal">

                            <el-input v-model="form.deal" type="textarea" />

                        </el-form-item>

                    </div>

                </el-form>

                <template #footer>

                    <span class="dialog-footer">

                        <el-button type="primary" @click="onSubmit" style="width:100px">确认</el-button>

                        <el-button @click="onCancel" style="width:100px">取消</el-button>

                    </span>

                </template>

            </el-dialog>

        </div>

    </div>

</template>

<script lang="ts">

export default {

    name: ""

}

</script>

<script setup lang="ts">

import { ref, reactive } from 'vue'

import { ArrowRight } from '@element-plus/icons-vue'

import { ElMessage, FormRules, ElMessageBox } from 'element-plus'

import axios from '../../http'

interface IPubilc {

    [k: string]: any

}

const userData = ref<IPubilc[]>([])

axios.get('/getinfo.json', { params: { page: 1, pageSize: 10000 } }).then(res => {

    userData.value = res.data.result

})

const tableData = ref([])

const currentPage = ref(1)

const pageSize = ref(7)

const totalPage = ref(1)

const handleSizeChange = (evt: any) => {

    pageSize.value = evt

    getData()

}

const handleCurrentChange = (evt: any) => {

    currentPage.value = evt

    getData()

}

const getData = () => {

    axios.get('/getcase.json', { params: { page: currentPage.value, pageSize: pageSize.value } }).then(res => {

        tableData.value = res.data.result

        totalPage.value = res.data.total

    })

}

getData()

const isEdit = ref(false)

const dialogshow = ref(false)

const edit = (row: any) => {

    dialogshow.value = true;

    form.value = JSON.parse(JSON.stringify(row))

    isEdit.value = true

}

const del = (row: any) => {

    ElMessageBox.confirm(`确定删除【当前经历】吗？`, {

        confirmButtonText: '确认',

        cancelButtonText: '取消',

    }).then(() => {

        axios.get('/deletecase.json', { params: { id: row.id, userid: row.userid } }).then((res: any) => {

            getData()

            ElMessage({

                type: 'success',

                message: res.data.message,

            })

        })

    })

}

const add = () => {

    dialogshow.value = true;

    form.value = JSON.parse(JSON.stringify(defaultForm))

    isEdit.value = false

}

const formRef = ref(null)

const onCancel = () => {

    dialogshow.value = false

    setTimeout(() => {

        form.value = JSON.parse(JSON.stringify(defaultForm))

    }, 100)

}

const onSubmit = () => {

    (formRef.value as any).validate((valid: any, fields: any) => {

        if (valid) {

            let url

            if (isEdit.value) {

                url = '/editcase.json'

            } else {

                url = '/addcase.json'

            }

            axios.post(url, {

                ...JSON.parse(JSON.stringify(form.value))

            }).then((res: any) => {

                getData()

                ElMessage({

                    type: 'success',

                    message: res.data.message

                })

                form.value = JSON.parse(JSON.stringify(defaultForm))

                dialogshow.value = false

                setTimeout(() => {

                    (formRef.value as any).resetFields()

                }, 100)

            })

        }

    })

}

const defaultForm = {

    userid: 1,

    name: '',

    year: '',

    type: '',

    intro: '',

    error: '',

    deal: ''

}

const form = ref({

    userid: 1,

    name: '',

    year: '',

    type: '',

    intro: '',

    error: '',

    deal: ''

})

const type = ref([

    { value: 0, message: '专利' },

    { value: 1, message: '商标' },

    { value: 2, message: '版权' },

])

const rules = reactive<FormRules>({

    name: [{ required: true, message: '请填写案例名称', trigger: 'blur' }],

    year: [{ required: true, message: '请选择案例年份', trigger: 'blur' }],

    type: [{ required: true, message: '请选择案例所属业务类型', trigger: 'change' }],

    intro: [{ required: true, message: '请填写案例基本情况简介', trigger: 'change' }],

})

const tableRowClassName = ({

    row,

    rowIndex,

}: {

    row: any

    rowIndex: any

}) => {

    if ((rowIndex + 1) % 2 === 0) {

        // return 'warning-row'

        return 'success-row'

    }

    // else if (rowIndex % 3 === 0) {

    //     return 'success-row'

    // }

    return ''

}

</script>

<style lang="less" scoped>

.case {

    width: 100%;

}

.form-wrap {

    width: 100%;

    display: flex;

    flex-direction: column;

    justify-content: space-around;

    padding: 0 50px;

    .btn-group {

        margin-bottom: 20px;

    }

    .el-table {

        height: 360px;

        overflow-y: auto;

    }

    .el-pagination {

        align-self: flex-end;

        padding: 10px 0;

    }

}

:deep(.el-dialog .el-dialog\_\_header .el-dialog\_\_headerbtn) {

    top: -5px;

}

.el-form {

    width: 550px;

    .el-form-item {

        margin-top: 30px;

    }

    .bottom {

        display: flex;

    }

    .date :deep(.el-input) {

        width: 100%;

        .el-input\_\_wrapper {

            width: 100%

        }

    }

}

</style>

<template>

    <div class="chat">

        <el-breadcrumb :separator-icon="ArrowRight">

            <el-breadcrumb-item :to="{ path: '/' }">主页</el-breadcrumb-item>

            <el-breadcrumb-item>在线互动沟通</el-breadcrumb-item>

        </el-breadcrumb>

        <div class="search">

            <el-form :model="form" ref="formRef" label-width="80px">

                <el-form-item label="关键词">

                    <el-input v-model="form.name" style="width: 200px;" />

                </el-form-item>

                <el-form-item label="工作年限">

                    <el-input-number v-model="form.years" :min="1" :max="30" />

                </el-form-item>

                <el-form-item label="业务类型" prop="type">

                    <el-select v-model="form.type" placeholder="请选择所属业务类型" style="width:100%">

                        <el-option v-for="(item,index) in type" :label="item.message" :value="item.value" />

                    </el-select>

                </el-form-item>

                <div class="btn">

                    <el-button type="primary" @click="onSubmit">查询</el-button>

                    <el-button @click="onReset">重置</el-button>

                </div>

            </el-form>

        </div>

        <div class="com-list">

            <div class="user" v-for="item in comList" :key="item.id" @click="communicate(item)">

                <img :src="item.avatar" alt="">

                <div class="info">

                    <div>{{item.name}}</div>

                    <div>工作年限：<span>{{item.years}}年</span></div>

                    <div>个人简介：<span>{{item.remark}}</span></div>

                </div>

                <p :class="['tag',`tag${item.status}`]">

                    {{status(item)}}

                </p>

            </div>

        </div>

        <el-dialog v-model="chat" :title="title" width="40%">

            <div class="chat-box" ref="chatRef">

                <div class="chat-item" v-for="(item,index) in chatMsg" :key="index">

                    <div class="ta" v-if="item.from===0">

                        <div class="avatar">TA</div>

                        <div class="msg">{{item.content}}</div>

                    </div>

                    <div class="wo" v-else>

                        <div class="msg">{{item.content}}</div>

                        <div class="avatar">我</div>

                    </div>

                </div>

            </div>

            <div class="chat-input">

                <el-input v-model="inputMsg" type="textarea" resize="none" @keydown.prevent.enter="send" />

                <el-button class="btn" type="primary" @click="send" style="width: 100px;">发送</el-button>

            </div>

        </el-dialog>

    </div>

</template>

<script lang="ts">

export default {

    name: ""

}

</script>

<script setup lang="ts">

import { ref, reactive, nextTick } from 'vue'

import { ArrowRight } from '@element-plus/icons-vue'

const chatRef = ref<Element | null>(null)

const inputMsg = ref('')

interface Imsg {

    content: string,

    from: number

}

const form = ref({

    name: '',

    type: '',

    years: 2

})

const type = ref([

    { value: 0, message: '专利' },

    { value: 1, message: '商标' },

    { value: 2, message: '版权' },

])

const onSubmit = () => { }

const onReset = () => {

    form.value = {

        name: '',

        type: '',

        years: 2

    }

}

const send = () => {

    if (inputMsg.value.trim() !== '') {

        let msg: Imsg = {

            content: inputMsg.value,

            from: 1

        }

        chatMsg.value.push(msg)

        inputMsg.value = ''

        if (chatRef.value) {

            nextTick(() => {

                if (chatRef.value)

                    chatRef.value.scrollTop = chatRef.value.clientHeight

            })

        }

    }

}

const chat = ref(false)

const chatMsg = ref<Imsg[]>([])

const title = ref('')

const communicate = (item: any) => {

    chat.value = true

    title.value = item.name

    chatMsg.value = item.msg

    nextTick(() => {

        if (chatRef.value)

            chatRef.value.scrollTop = chatRef.value.clientHeight

    })

}

const status = (item: any) => {

    let sta

    if (item.status === 0) {

        sta = '离线'

    } else if (item.status === 1) {

        sta = '在线'

    } else {

        sta = '忙碌'

    }

    return sta

}

interface Icom {

    id: number,

    name: string,

    avatar: string,

    status: number,

    msg: Imsg[],

    years: number,

    remark: string

}

const comList = ref<Icom[]>([])

</script>

<style lang="less" scoped>

.chat {

    width: 100%

}

.search {

    padding: 0 50px;

    .el-form {

        display: flex;

        justify-content: space-between;

    }

}

.com-list {

    margin-top: 40px;

    display: flex;

    align-items: center;

    flex-wrap: wrap;

    .user {

        position: relative;

        display: flex;

        align-items: center;

        height: 150px;

        width: 450px;

        background: #f0f0f0;

        border-radius: 10px;

        margin-left: 50px;

        margin-bottom: 30px;

        padding: 20px;

        cursor: pointer;

        transition: all 0.2s;

        &:hover {

            transform: scale(1.1);

        }

        img {

            height: 100px;

            width: 100px;

            border-radius: 50%;

            margin-right: 10px;

        }

        .info {

            font-size: 15px;

            div:nth-child(1) {

                font-weight: bold;

            }

            div:not(:nth-child(1)) {

                color: gray;

            }

            span {

                font-weight: bold;

            }

            div:nth-child(3) {

                display: -webkit-box;

                -webkit-box-orient: vertical;

                overflow: hidden;

                -webkit-line-clamp: 3;

            }

        }

        .tag {

            position: absolute;

            right: 0;

            top: 0;

            color: rgb(15, 221, 60);

            width: 80px;

            height: 30px;

            background: linear-gradient(to right, #fbd3e9, #bb377d);

            border-radius: 0 10px 0 10px;

            line-height: 30px;

            text-align: center;

            &.tag0 {

                color: #ccc;

            }

            &.tag2 {

                color: rgb(235, 195, 18);

            }

        }

    }

}

.chat-box {

    max-height: 300px;

    overflow-y: auto;

    background: #f0f0f0;

    padding: 10px;

    .chat-item {

        margin-bottom: 10px;

        .avatar {

            width: 50px;

            font-size: 30px;

            height: 50px;

            color: #fff;

            font-weight: bold;

            line-height: 50px;

            text-align: center;

            border-radius: 50%;

        }

        .msg {

            min-height: 50px;

            line-height: 50px;

            margin: 0 10px;

            padding: 0 30px;

            background: skyblue;

            border-radius: 25px;

        }

        .ta {

            .msg {

                background: pink;

            }

        }

        .ta {

            display: flex;

            align-items: center;

            justify-content: flex-start;

            .avatar {

                background: #bb377d;

            }

        }

        .wo {

            display: flex;

            align-items: center;

            justify-content: flex-end;

            .avatar {

                background: indigo;

            }

        }

    }

}

.chat-input {

    display: flex;

    flex-direction: column;

    .btn {

        align-self: flex-end;

    }

}

/deep/.el-overlay .el-overlay-dialog .el-dialog .el-dialog\_\_body {

    padding: 0;

    padding-bottom: 10px;

}

</style>

<template>

    <div class="query">

        <el-breadcrumb :separator-icon="ArrowRight">

            <el-breadcrumb-item :to="{ path: '/' }">主页</el-breadcrumb-item>

            <el-breadcrumb-item>信息检索</el-breadcrumb-item>

        </el-breadcrumb>

        <div class="search">

            <el-form :model="form" ref="formRef" label-width="80px">

                <el-form-item label="关键词">

                    <el-input v-model="form.name" style="width: 200px;" />

                </el-form-item>

                <el-form-item label="工作年限">

                    <el-input-number v-model="form.years" :min="1" :max="30" />

                </el-form-item>

                <el-form-item label="业务类型" prop="type">

                    <el-select v-model="form.type" placeholder="请选择所属业务类型" style="width:100%">

                        <el-option v-for="(item,index) in type" :label="item.message" :value="item.value" />

                    </el-select>

                </el-form-item>

                <div class="btn">

                    <el-button type="primary" @click="onSubmit">查询</el-button>

                    <el-button @click="onReset">重置</el-button>

                </div>

            </el-form>

        </div>

        <div class="table">

            <el-table :data="tableData" style="width: 100%" :row-class-name="tableRowClassName">

                <el-table-column prop="id" label="编号" width="55px" />

                <el-table-column prop="name" label="姓名" width="80px" />

                <el-table-column prop="gender" label="性别" width="55px">

                    <template #default="{ row, column, $index }">

                        <span>

                            {{row.gender===1?'男':'女'}}

                        </span>

                    </template>

                </el-table-column>

                <el-table-column prop="educational" label="学历" width="120px">

                    <template #default="{ row, column, $index }">

                        <span>

                            {{row.educational===0?'初中及以下':row.educational===1?'高中':row.educational===2?'统招专科':row.educational===3?'统招本科':row.educational===4?'硕士及以上':"其他"}}

                        </span>

                    </template>

                </el-table-column>

                <el-table-column prop="unit" label="单位名称" />

                <el-table-column prop="phone" label="联系电话" />

                <el-table-column label="操作" width="120px">

                    <template #default="{ row, column, $index }">

                        <el-button link type="primary" size="small" @click="check(row)">查看详情</el-button>

                    </template>

                </el-table-column>

            </el-table>

            <el-pagination v-model:currentPage="currentPage" v-model:page-size="pageSize" :page-sizes="[7, 14, 21, 28]"

                background layout="total, sizes, prev, pager, next, jumper" :total="totalPage"

                @size-change="handleSizeChange" @current-change="handleCurrentChange" />

            <el-dialog v-model="dialogshow" width="580px" destroy-on-close>

                <div class="dialog-wrap" ref="dialog">

                    <div class="self">

                        <div class="c-title">个人信息</div>

                        <div class="con">

                            <div class="con-item">

                                <span>姓名：</span>

                                <div>{{info.name}}</div>

                            </div>

                            <div class="con-item">

                                <span>性别：</span>

                                <div>{{info.gender===1?"男":'女'}}</div>

                            </div>

                            <div class="con-item">

                                <span>学历：</span>

                                <div>

                                    {{info.educational===0?'初中及以下':info.educational===1?'高中':info.educational===2?'统招专科':info.educational===3?'统招本科':info.educational===4?'硕士及以上':"其他"}}

                                </div>

                            </div>

                            <div class="con-item">

                                <span>专业：</span>

                                <div>{{info.major}}</div>

                            </div>

                            <div class="con-item" v-if="info.remark">

                                <span>个人简介：</span>

                                <div>{{info.remark}}</div>

                            </div>

                        </div>

                    </div>

                    <br>

                    <hr>

                    <br>

                    <div class="compan">

                        <div class="c-title">企业信息</div>

                        <div class="con">

                            <div class="con-item">

                                <span>单位名称：</span>

                                <div>{{info.unit}}</div>

                            </div>

                            <div class="con-item">

                                <span>单位地址：</span>

                                <div>{{info.address}}</div>

                            </div>

                            <div class="con-item">

                                <span>联系人：</span>

                                <div>{{info.linkman}}</div>

                            </div>

                            <div class="con-item">

                                <span>联系电话：</span>

                                <div>{{info.phone}}</div>

                            </div>

                        </div>

                    </div>

                    <br>

                    <hr>

                    <br>

                    <div class="exp" v-if="info.record.length!==0">

                        <div class="c-title">工作履历</div>

                        <div class="con" v-for="(item,index) in info.record" :key="index">

                            <div class="con-item">

                                <span>工作时间：</span>

                                <div>{{item.time[0]+'至'+item.time[1]}}</div>

                            </div>

                            <div class="con-item">

                                <span>公司名称：</span>

                                <div>{{item.company}}</div>

                            </div>

                            <div class="con-item">

                                <span>职位名称：</span>

                                <div>{{item.position}}</div>

                            </div>

                            <br>

                        </div>

                    </div>

                    <br v-if="info.record.length!==0">

                    <hr v-if="info.record.length!==0">

                    <br v-if="info.record.length!==0">

                    <div class="train" v-if="info.train.length!==0">

                        <div class="c-title">继续教育</div>

                        <div class="con" v-for="(item,index) in info.train" :key="index">

                            <div class="con-item">

                                <span>培训时间：</span>

                                <div>{{item.time[0]+'至'+item.time[1]}}</div>

                            </div>

                            <div class="con-item">

                                <span>机构名称：</span>

                                <div>{{item.company}}</div>

                            </div>

                            <div class="con-item">

                                <span>培训内容：</span>

                                <div>{{item.position}}</div>

                            </div>

                            <br>

                        </div>

                    </div>

                    <br v-if="info.train.length!==0">

                    <hr v-if="info.train.length!==0">

                    <br v-if="info.train.length!==0">

                    <div class="case" v-if="info.case.length!==0">

                        <div class="c-title">经典案例</div>

                        <div class="con" v-for="(item,index) in info.case" :key="index">

                            <div class="con-item">

                                <span>案例名称：</span>

                                <div>{{item.name}}</div>

                            </div>

                            <div class="con-item">

                                <span>案例年份：</span>

                                <div>{{item.year}}</div>

                            </div>

                            <div class="con-item">

                                <span>所属业务类型：</span>

                                <div>{{info.type===0?'专利':info.type===1?'商标':'版权'}}</div>

                            </div>

                            <div class="con-item">

                                <span>案例基本情况：</span>

                                <div>{{item.intro}}</div>

                            </div>

                            <div class="con-item" v-if="item.error">

                                <span>问题争议点：</span>

                                <div>{{item.error}}</div>

                            </div>

                            <div class="con-item" v-if="item.deal">

                                <span>方案解决：</span>

                                <div>{{item.deal}}</div>

                            </div>

                            <br>

                        </div>

                    </div>

                </div>

            </el-dialog>

        </div>

    </div>

</template>

<script lang="ts">

export default {

    name: ""

}

</script>

<script setup lang="ts">

import { ref, reactive, nextTick } from 'vue'

import { ArrowRight } from '@element-plus/icons-vue'

import axios from '../../http'

interface IPubilc {

    [k: string]: any

}

const currentPage = ref(1)

const pageSize = ref(7)

const totalPage = ref(1)

const tableData = ref([])

const handleSizeChange = (evt: any) => {

    pageSize.value = evt

    getInfo()

}

const handleCurrentChange = (evt: any) => {

    currentPage.value = evt

    getInfo()

}

const getInfo = () => {

    axios.get('/getinfo.json', { params: { page: currentPage.value, pageSize: pageSize.value } }).then(res => {

        totalPage.value = res.data.total

        tableData.value = res.data.result

    })

}

getInfo()

const dialogshow = ref(false)

const info = ref<IPubilc>({})

const dialog = ref(null)

const check = (row: any) => {

    setTimeout(() => {

        info.value = JSON.parse(JSON.stringify(row))

        dialogshow.value = true

    }, 80)

}

const form = ref({

    name: '',

    type: '',

    years: 2

})

const type = ref([

    { value: 0, message: '专利' },

    { value: 1, message: '商标' },

    { value: 2, message: '版权' },

])

const onSubmit = () => { }

const onReset = () => {

    form.value = {

        name: '',

        type: '',

        years: 2

    }

}

const tableRowClassName = ({

    row,

    rowIndex,

}: {

    row: any

    rowIndex: any

}) => {

    if ((rowIndex + 1) % 2 === 0) {

        // return 'warning-row'

        return 'success-row'

    }

    // else if (rowIndex % 3 === 0) {

    //     return 'success-row'

    // }

    return ''

}

</script>

<style lang="less" scoped>

.query {

    width: 100%;

    .search {

        padding: 0 50px;

        .el-form {

            display: flex;

            justify-content: space-between;

        }

    }

    .table {

        display: flex;

        flex-direction: column;

        padding: 0 50px;

        margin-top: 30px;

        .el-table {

            height: 360px;

            overflow-y: auto;

        }

        .el-pagination {

            align-self: flex-end;

            padding: 10px 0;

        }

    }

}

.dialog-wrap {

    padding: 20px;

    padding-left: 30px;

    max-height: 450px;

    overflow-y: auto;

    .c-title {

        font-weight: bold;

        font-size: 17px;

        margin: 10px 0;

    }

    .con {

        .con-item {

            display: flex;

            span {

                font-weight: bold;

                color: gray;

            }

            div {

                width: 400px;

            }

        }

    }

}

</style>

<template>

    <div class="setting">

        <div class="form-wrap">

            <h3>更改管理员信息</h3>

            <el-form ref="formRef" :model="form" :rules="rules" label-width="100px" class="demo-form">

                <el-form-item label="用户名" prop="username">

                    <el-input v-model="form.username" />

                </el-form-item>

                <el-form-item label="新密码" prop="password" style="margin-top: 30px;">

                    <el-input v-model="form.password" type="password" show-password />

                </el-form-item>

                <el-form-item label="确认新密码" prop="cpassword" style="margin-top: 30px;">

                    <el-input v-model="form.cpassword" type="password" show-password />

                </el-form-item>

                <el-button @click="handleLogin" type="primary"

                    style="margin-top: 20px;width: 400px;margin-left: 100px;">

                    确定</el-button>

            </el-form>

        </div>

    </div>

</template>

<script lang="ts">

export default {

    name: ""

}

</script>

<script lang="ts" setup>

import { ref, reactive } from 'vue'

import { useRouter } from 'vue-router'

import { ElMessage, FormInstance, FormRules } from 'element-plus'

const router = useRouter()

const form = reactive({

    username: '',

    password: '',

    cpassword: ''

})

const rules = reactive<FormRules>({

    username: [

        {

            trigger: 'blur',

            required: true,

            validator: (rule, value, callback) => {

                var passwordreg = /^\w+$/

                if (!passwordreg.test(value)) {

                    callback(

                        new Error(

                            '用户名必须英文字母、数字、下划线组成!'

                        )

                    )

                } else {

                    callback()

                }

            }

        },

        { min: 5, max: 8, message: '请输入5-8位用户名', trigger: 'blur' },

    ],

    password: [

        {

            trigger: 'blur',

            required: true,

            validator: (rule, value, callback) => {

                var passwordreg = /^\w+$/

                if (!passwordreg.test(value)) {

                    callback(

                        new Error(

                            '用户名必须英文字母、数字、下划线组成!'

                        )

                    )

                } else {

                    callback()

                }

            }

        },

        { min: 6, max: 10, message: '请输入6-10位用户密码', trigger: 'blur' },

    ],

    cpassword: [

        {

            trigger: 'blur',

            required: true,

            validator: (rule, value, callback) => {

                var passwordreg = /^\w+$/

                if (!passwordreg.test(value)) {

                    callback(

                        new Error(

                            '用户名必须英文字母、数字组成!'

                        )

                    )

                } else {

                    callback()

                }

            }

        },

        { min: 6, max: 10, message: '请输入6-10位用户密码', trigger: 'blur' },

    ],

})

const formRef = ref(null)

const handleLogin = () => {

    (formRef.value as any).validate((valid: any, fields: any) => {

        if (valid) {

            if (form.password === form.cpassword) {

                localStorage.setItem('newuserinfo', JSON.stringify({ ...form }))

                router.push({ name: 'login' })

                ElMessage({

                    type: 'success',

                    message: '修改用户信息成功，请重新登录'

                })

            } else {

                ElMessage({

                    type: 'warning',

                    message: '请保证密码一致'

                })

            }

        }

    })

}

</script>

<style lang="less" scoped>

.setting {

    width: 100%;

    user-select: none;

    display: flex;

    flex-direction: column;

    align-items: center;

    .form-wrap {

        box-shadow: 2px 2px 10px #06C;

        border: 1px solid #ccc;

        border-radius: 10px;

        padding: 20px;

        margin-top: 50px;

    }

    h3 {

        margin: 50px 0 30px 50px;

        margin-top: 0;

    }

}

</style>

import { app, BrowserWindow, shell, ipcMain, globalShortcut, Menu, Tray } from 'electron'

import { release } from 'os'

import { join } from 'path'

// Disable GPU Acceleration for Windows 7

if (release().startsWith('6.1')) app.disableHardwareAcceleration()

// Set application name for Windows 10+ notifications

if (process.platform === 'win32') app.setAppUserModelId(app.getName())

if (!app.requestSingleInstanceLock()) {

  app.quit()

  process.exit(0)

}

process.env['ELECTRON\_DISABLE\_SECURITY\_WARNINGS'] = 'true'

let win: BrowserWindow | null = null

let tray: Tray | null = null

const contextMenu = Menu.buildFromTemplate([{

  label: '退出系统',

  click: function () {

    tray?.destroy()

    app.quit()

  }

}])

async function createWindow() {

  win = new BrowserWindow({

    title: '',

    width: 1280,

    height: 720,

    // transparent: true,

    // backgroundColor: '#00000000',

    webPreferences: {

      preload: join(\_\_dirname, '../preload/index.cjs'),

      nodeIntegration: true,

      contextIsolation: false,

    },

  })

  // win.webContents.openDevTools()

  // globalShortcut.register('CommandOrControl+q', function () {

  //   win?.webContents.openDevTools()

  // })

  win.setMenu(null)

  //设置托盘

  if (process.env.NODE\_ENV === 'development') {

    tray = new Tray('./dist/renderer/icon.ico')

  } else {

    tray = new Tray(join(\_\_dirname, '../renderer/icon.ico'))

  }

  tray.setToolTip('xx系统')

  tray.setContextMenu(contextMenu)

  tray.on('click', () => {

    win?.show()

    win?.focus()

    win?.setSkipTaskbar(false)

  })

  if (app.isPackaged) {

    win.loadFile(join(\_\_dirname, '../renderer/index.html'))

  } else {

    const url = `http://${process.env['VITE\_DEV\_SERVER\_HOST']}:${process.env['VITE\_DEV\_SERVER\_PORT']}`

    win.loadURL(url)

  }

  win.webContents.on('did-finish-load', () => {

    win?.webContents.send('main-process-message', new Date().toLocaleString())

  })

  win.webContents.setWindowOpenHandler(({ url }) => {

    if (url.startsWith('https:')) shell.openExternal(url)

    return { action: 'deny' }

  })

}

app.whenReady().then(createWindow)

app.on('window-all-closed', () => {

  win = null

  if (process.platform !== 'darwin') app.quit()

})

app.on('second-instance', () => {

  if (win) {

    if (win.isMinimized()) win.restore()

    win.focus()

  }

})

app.on('activate', () => {

  const allWindows = BrowserWindow.getAllWindows()

  if (allWindows.length) {

    allWindows[0].focus()

  } else {

    createWindow()

  }

})

ipcMain.handle("open-win", (event, arg) => {

  const childWindow = new BrowserWindow({

    webPreferences: {

      preload: join(\_\_dirname, "../preload/index.cjs"),

    },

  });

  if (app.isPackaged) {

    childWindow.loadFile(join(\_\_dirname, `../renderer/index.html`), {

      hash: `${arg}`,

    })

  } else {

    const url = `http://${process.env["VITE\_DEV\_SERVER\_HOST"]}:${process.env["VITE\_DEV\_SERVER\_PORT"]}/#${arg}`

    childWindow.loadURL(url);

  }

});

import Mock, { Random } from 'mockjs'

interface IPubilc {

    [k: string]: any

}

let info: IPubilc[] = []

const handlePage = (data: any[], page: any, pageSize: any) => {

    let proportion = pageSize;

    let num = 0;

    let \_data = [];

    for (let i = 0; i < data.length; i++) {

        if (i % proportion == 0 && i != 0) {

            \_data.push(data.slice(num, i));

            num = i;

        }

        if ((i + 1) == data.length) {

            \_data.push(data.slice(num, (i + 1)));

        }

    }

    return \_data[page - 1];

}

export default [

    //基本信息

    {

        url: '/getinfo.json',

        method: 'get',

        response(option: any) {

            const { page, pageSize } = option.query

            return {

                result: handlePage(info, page, pageSize),

                total: info.length,

                totalPage: Math.ceil(info.length / pageSize)

            }

        }

    },

    {

        url: '/addinfo.json',

        method: 'post',

        response(option: any) {

            info.unshift({

                id: info.length + 1,

                ...option.body,

                record: [],

                train: [],

                case: []

            })

            return {

                status: 200,

                message: '添加成功'

            }

        }

    },

    {

        url: '/editinfo.json',

        method: 'post',

        response(option: any) {

            const { id, name, educational, major, unit, address, linkman, phone, gender } = option.body

            let infoItem = info.find(item => item.id === id)

            if (infoItem) {

                infoItem.name = name

                infoItem.educational = educational

                infoItem.major = major

                infoItem.unit = unit

                infoItem.address = address

                infoItem.linkman = linkman

                infoItem.phone = phone

                infoItem.gender = gender

            }

            return {

                status: 200,

                message: '编辑成功'

            }

        }

    },

    {

        url: '/deleteinfo.json',

        method: 'get',

        response(option: any) {

            const { id } = option.query

            info = info.filter(item => !id.includes(item.id + ''))

            return {

                status: 200,

                message: '删除成功'

            }

        }

    },

    //工作履历

    {

        url: '/getrecord.json',

        method: 'get',

        response(option: any) {

            const { page, pageSize } = option.query

            let record = info.map(item => {

                return item.record

            })

                .reduce((pre, cur) => {

                    return pre.concat(cur)

                }, [])

            return {

                result: handlePage(record, page, pageSize),

                total: record.length,

                totalPage: Math.ceil(record.length / pageSize)

            }

        }

    },

    {

        url: '/addrecord.json',

        method: 'post',

        response(option: any) {

            const { userid } = option.body

            let infoItem = info.find(item => item.id === userid)

            if (infoItem) {

                infoItem.record.unshift({

                    id: `${userid}-${infoItem.record.length + 1}`,

                    ...option.body

                })

            }

            return {

                status: 200,

                message: '添加成功'

            }

        }

    },

    {

        url: '/editrecord.json',

        method: 'post',

        response(option: any) {

            const { userid, id, time, company, position } = option.body

            let infoItem = info.find(item => item.id === userid)

            let recordItem

            if (infoItem) {

                recordItem = infoItem.record.find((item: any) => item.id === id)

            }

            if (recordItem) {

                recordItem.time = time

                recordItem.company = company

                recordItem.position = position

            }

            return {

                status: 200,

                message: '编辑成功'

            }

        }

    },

    {

        url: '/deleterecord.json',

        method: 'get',

        response(option: any) {

            const { id, userid } = option.query

            let infoItem = info.find(item => item.id === JSON.parse(userid))

            if (infoItem) {

                infoItem.record = infoItem.record.filter((item: any) => !id.includes(item.id + ''))

            }

            return {

                status: 200,

                message: '删除成功'

            }

        }

    },

    //继续教育

    {

        url: '/gettrain.json',

        method: 'get',

        response(option: any) {

            const { page, pageSize } = option.query

            let train = info.map(item => {

                return item.train

            })

                .reduce((pre, cur) => {

                    return pre.concat(cur)

                }, [])

            return {

                result: handlePage(train, page, pageSize),

                total: train.length,

                totalPage: Math.ceil(train.length / pageSize)

            }

        }

    },

    {

        url: '/addtrain.json',

        method: 'post',

        response(option: any) {

            const { userid } = option.body

            let infoItem = info.find(item => item.id === userid)

            if (infoItem) {

                infoItem.train.unshift({

                    id: `${userid}-${infoItem.train.length + 1}`,

                    ...option.body

                })

            }

            return {

                status: 200,

                message: '添加成功'

            }

        }

    },

    {

        url: '/edittrain.json',

        method: 'post',

        response(option: any) {

            const { userid, id, time, company, position } = option.body

            let infoItem = info.find(item => item.id === userid)

            let trainItem

            if (infoItem) {

                trainItem = infoItem.train.find((item: any) => item.id === id)

            }

            if (trainItem) {

                trainItem.time = time

                trainItem.company = company

                trainItem.position = position

            }

            return {

                status: 200,

                message: '编辑成功'

            }

        }

    },

    {

        url: '/deletetrain.json',

        method: 'get',

        response(option: any) {

            const { id, userid } = option.query

            let infoItem = info.find(item => item.id === JSON.parse(userid))

            if (infoItem) {

                infoItem.train = infoItem.train.filter((item: any) => !id.includes(item.id + ''))

            }

            return {

                status: 200,

                message: '删除成功'

            }

        }

    },

    //经典案例

    {

        url: '/getcase.json',

        method: 'get',

        response(option: any) {

            const { page, pageSize } = option.query

            let ca = info.map(item => {

                return item.case

            })

                .reduce((pre, cur) => {

                    return pre.concat(cur)

                }, [])

            return {

                result: handlePage(ca, page, pageSize),

                total: ca.length,

                totalPage: Math.ceil(ca.length / pageSize)

            }

        }

    },

    {

        url: '/addcase.json',

        method: 'post',

        response(option: any) {

            const { userid } = option.body

            let infoItem = info.find(item => item.id === userid)

            if (infoItem) {

                infoItem.case.unshift({

                    id: `${userid}-${infoItem.case.length + 1}`,

                    ...option.body

                })

            }

            return {

                status: 200,

                message: '添加成功'

            }

        }

    },

    {

        url: '/editcase.json',

        method: 'post',

        response(option: any) {

            const { userid, id, name, year, type, intro, error, deal } = option.body

            let infoItem = info.find(item => item.id === JSON.parse(userid))

            let caseItem

            if (infoItem) {

                caseItem = infoItem.case.find((item: any) => item.id === id)

            }

            if (caseItem) {

                caseItem.name = name

                caseItem.year = year

                caseItem.type = type

                caseItem.intro = intro

                caseItem.error = error

                caseItem.deal = deal

            }

            return {

                status: 200,

                message: '编辑成功'

            }

        }

    },

    {

        url: '/deletecase.json',

        method: 'get',

        response(option: any) {

            const { id, userid } = option.query

            let infoItem = info.find(item => item.id === JSON.parse(userid))

            if (infoItem) {

                infoItem.case = infoItem.case.filter((item: any) => !id.includes(item.id + ''))

            }

            return {

                status: 200,

                message: '删除成功'

            }

        }

    }

]

import { defineConfig } from 'vite'

import vue from '@vitejs/plugin-vue'

import resolve, { lib2esm } from 'vite-plugin-resolve'

import electron from 'vite-plugin-electron/renderer'

import pkg from '../../package.json'

import { resolve as resolves } from 'path'

import { viteMockServe } from 'vite-plugin-mock'

import AutoImport from 'unplugin-auto-import/vite'

import Components from 'unplugin-vue-components/vite'

import { ElementPlusResolver } from 'unplugin-vue-components/resolvers'

import polyfillExports from 'vite-plugin-electron/polyfill-exports'

export default defineConfig({

  mode: process.env.NODE\_ENV,

  root: \_\_dirname,

  resolve: {

    alias: {

      '@': '/src/',

    }

  },

  css: {

    preprocessorOptions: {

      less: {

        charset: false,

        additionalData: '@import "/src/assets/css/global.less";',

      },

    },

  },

  plugins: [

    polyfillExports(),

    vue(),

    electron(),

    AutoImport({

      resolvers: [ElementPlusResolver()],

    }),

    Components({

      resolvers: [ElementPlusResolver()],

    }),

    resolve(

      {

        'electron-store': 'export default require("electron-store");',

        sqlite3: lib2esm('sqlite3', { format: 'cjs' }),

        serialport: lib2esm(

          'serialport',

          [

            'SerialPort',

            'SerialPortMock',

          ],

          { format: 'cjs' },

        ),

      }

    ),

    viteMockServe({

      mockPath: "./src/mock/source", // 解析，路径可根据实际变动

      localEnabled: false, // 开发环境

      prodEnabled: true, // 生产环境设为true，也可以根据官方文档格式

      injectCode:

        ` import { setupProdMockServer } from './src/mock';

        setupProdMockServer(); `,

      watchFiles: true, // 监听文件内容变更

      injectFile: resolves("src/main.ts"), // 在main.ts注册后需要在此处注入，否则可能报找不到setupProdMockServer的错误

    })

  ],

  base: './',

  build: {

    outDir: '../../dist/renderer',

    emptyOutDir: true,

    sourcemap: false,

    minify: 'terser',

    cssCodeSplit: false,

    chunkSizeWarningLimit: 1500,

  },

  server: {

    host: pkg.env.VITE\_DEV\_SERVER\_HOST,

    port: pkg.env.VITE\_DEV\_SERVER\_PORT,

  },

})

var Handler = require('./mock/handler')

var Util = require('./mock/util')

var Random = require('./mock/random')

var RE = require('./mock/regexp')

var toJSONSchema = require('./mock/schema')

var valid = require('./mock/valid')

var XHR

if (typeof window !== 'undefined') XHR = require('./mock/xhr')

var Mock = {

Handler: Handler,

Random: Random,

Util: Util,

XHR: XHR,

RE: RE,

toJSONSchema: toJSONSchema,

valid: valid,

heredoc: Util.heredoc,

setup: function(settings) {

return XHR.setup(settings)

},

\_mocked: {}

}

Mock.version = '1.0.1-beta3'

// 避免循环依赖

if (XHR) XHR.Mock = Mock

/\*

\* Mock.mock( template )

\* Mock.mock( function() )

\* Mock.mock( rurl, template )

\* Mock.mock( rurl, function(options) )

\* Mock.mock( rurl, rtype, template )

\* Mock.mock( rurl, rtype, function(options) )

根据数据模板生成模拟数据。

\*/

Mock.mock = function(rurl, rtype, template) {

// Mock.mock(template)

if (arguments.length === 1) {

return Handler.gen(rurl)

}

// Mock.mock(rurl, template)

if (arguments.length === 2) {

template = rtype

rtype = undefined

}

// 拦截 XHR

if (XHR) window.XMLHttpRequest = XHR

Mock.\_mocked[rurl + (rtype || '')] = {

rurl: rurl,

rtype: rtype,

template: template

}

return Mock

}

module.exports = Mock

var Constant = require('../constant')

var Util = require('../util')

var toJSONSchema = require('../schema')

function valid(template, data) {

var schema = toJSONSchema(template)

var result = Diff.diff(schema, data)

for (var i = 0; i < result.length; i++) {

// console.log(template, data)

// console.warn(Assert.message(result[i]))

}

return result

}

var Diff = {

diff: function diff(schema, data, name /\* Internal Use Only \*/ ) {

var result = []

// 先检测名称 name 和类型 type，如果匹配，才有必要继续检测

if (

this.name(schema, data, name, result) &&

this.type(schema, data, name, result)

) {

this.value(schema, data, name, result)

this.properties(schema, data, name, result)

this.items(schema, data, name, result)

}

return result

},

/\* jshint unused:false \*/

name: function(schema, data, name, result) {

var length = result.length

Assert.equal('name', schema.path, name + '', schema.name + '', result)

return result.length === length

},

type: function(schema, data, name, result) {

var length = result.length

switch (schema.type) {

case 'string':

// 跳过含有『占位符』的属性值，因为『占位符』返回值的类型可能和模板不一致，例如 '@int' 会返回一个整形值

if (schema.template.match(Constant.RE\_PLACEHOLDER)) return true

break

case 'array':

if (schema.rule.parameters) {

// name|count: array

if (schema.rule.min !== undefined && schema.rule.max === undefined) {

// 跳过 name|1: array，因为最终值的类型（很可能）不是数组，也不一定与 `array` 中的类型一致

if (schema.rule.count === 1) return true

}

// 跳过 name|+inc: array

if (schema.rule.parameters[2]) return true

}

break

case 'function':

// 跳过 `'name': function`，因为函数可以返回任何类型的值。

return true

}

Assert.equal('type', schema.path, Util.type(data), schema.type, result)

return result.length === length

},

value: function(schema, data, name, result) {

var length = result.length

var rule = schema.rule

var templateType = schema.type

if (templateType === 'object' || templateType === 'array' || templateType === 'function') return true

// 无生成规则

if (!rule.parameters) {

switch (templateType) {

case 'regexp':

Assert.match('value', schema.path, data, schema.template, result)

return result.length === length

case 'string':

// 同样跳过含有『占位符』的属性值，因为『占位符』的返回值会通常会与模板不一致

if (schema.template.match(Constant.RE\_PLACEHOLDER)) return result.length === length

break

}

Assert.equal('value', schema.path, data, schema.template, result)

return result.length === length

}

// 有生成规则

var actualRepeatCount

switch (templateType) {

case 'number':

var parts = (data + '').split('.')

parts[0] = +parts[0]

// 整数部分

// |min-max

if (rule.min !== undefined && rule.max !== undefined) {

Assert.greaterThanOrEqualTo('value', schema.path, parts[0], Math.min(rule.min, rule.max), result)

// , 'numeric instance is lower than the required minimum (minimum: {expected}, found: {actual})')

Assert.lessThanOrEqualTo('value', schema.path, parts[0], Math.max(rule.min, rule.max), result)

}

// |count

if (rule.min !== undefined && rule.max === undefined) {

Assert.equal('value', schema.path, parts[0], rule.min, result, '[value] ' + name)

}

// 小数部分

if (rule.decimal) {

// |dmin-dmax

if (rule.dmin !== undefined && rule.dmax !== undefined) {

Assert.greaterThanOrEqualTo('value', schema.path, parts[1].length, rule.dmin, result)

Assert.lessThanOrEqualTo('value', schema.path, parts[1].length, rule.dmax, result)

}

// |dcount

if (rule.dmin !== undefined && rule.dmax === undefined) {

Assert.equal('value', schema.path, parts[1].length, rule.dmin, result)

}

}

break

case 'boolean':

break

case 'string':

// 'aaa'.match(/a/g)

actualRepeatCount = data.match(new RegExp(schema.template, 'g'))

actualRepeatCount = actualRepeatCount ? actualRepeatCount.length : 0

// |min-max

if (rule.min !== undefined && rule.max !== undefined) {

Assert.greaterThanOrEqualTo('repeat count', schema.path, actualRepeatCount, rule.min, result)

Assert.lessThanOrEqualTo('repeat count', schema.path, actualRepeatCount, rule.max, result)

}

// |count

if (rule.min !== undefined && rule.max === undefined) {

Assert.equal('repeat count', schema.path, actualRepeatCount, rule.min, result)

}

break

case 'regexp':

actualRepeatCount = data.match(new RegExp(schema.template.source.replace(/^\^|\$$/g, ''), 'g'))

actualRepeatCount = actualRepeatCount ? actualRepeatCount.length : 0

// |min-max

if (rule.min !== undefined && rule.max !== undefined) {

Assert.greaterThanOrEqualTo('repeat count', schema.path, actualRepeatCount, rule.min, result)

Assert.lessThanOrEqualTo('repeat count', schema.path, actualRepeatCount, rule.max, result)

}

// |count

if (rule.min !== undefined && rule.max === undefined) {

Assert.equal('repeat count', schema.path, actualRepeatCount, rule.min, result)

}

break

}

return result.length === length

},

properties: function(schema, data, name, result) {

var length = result.length

var rule = schema.rule

var keys = Util.keys(data)

if (!schema.properties) return

// 无生成规则

if (!schema.rule.parameters) {

Assert.equal('properties length', schema.path, keys.length, schema.properties.length, result)

} else {

// 有生成规则

// |min-max

if (rule.min !== undefined && rule.max !== undefined) {

Assert.greaterThanOrEqualTo('properties length', schema.path, keys.length, Math.min(rule.min, rule.max), result)

Assert.lessThanOrEqualTo('properties length', schema.path, keys.length, Math.max(rule.min, rule.max), result)

}

// |count

if (rule.min !== undefined && rule.max === undefined) {

// |1, |>1

if (rule.count !== 1) Assert.equal('properties length', schema.path, keys.length, rule.min, result)

}

}

if (result.length !== length) return false

for (var i = 0; i < keys.length; i++) {

result.push.apply(

result,

this.diff(

function() {

var property

Util.each(schema.properties, function(item /\*, index\*/ ) {

if (item.name === keys[i]) property = item

})

return property || schema.properties[i]

}(),

data[keys[i]],

keys[i]

)

)

}

return result.length === length

},

items: function(schema, data, name, result) {

var length = result.length

if (!schema.items) return

var rule = schema.rule

// 无生成规则

if (!schema.rule.parameters) {

Assert.equal('items length', schema.path, data.length, schema.items.length, result)

} else {

// 有生成规则

// |min-max

if (rule.min !== undefined && rule.max !== undefined) {

Assert.greaterThanOrEqualTo('items', schema.path, data.length, (Math.min(rule.min, rule.max) \* schema.items.length), result,

'[{utype}] array is too short: {path} must have at least {expected} elements but instance has {actual} elements')

Assert.lessThanOrEqualTo('items', schema.path, data.length, (Math.max(rule.min, rule.max) \* schema.items.length), result,

'[{utype}] array is too long: {path} must have at most {expected} elements but instance has {actual} elements')

}

// |count

if (rule.min !== undefined && rule.max === undefined) {

// |1, |>1

if (rule.count === 1) return result.length === length

else Assert.equal('items length', schema.path, data.length, (rule.min \* schema.items.length), result)

}

// |+inc

if (rule.parameters[2]) return result.length === length

}

if (result.length !== length) return false

for (var i = 0; i < data.length; i++) {

result.push.apply(

result,

this.diff(

schema.items[i % schema.items.length],

data[i],

i % schema.items.length

)

)

}

return result.length === length

}

}

var Assert = {

message: function(item) {

return (item.message ||

'[{utype}] Expect {path}\'{ltype} {action} {expected}, but is {actual}')

.replace('{utype}', item.type.toUpperCase())

.replace('{ltype}', item.type.toLowerCase())

.replace('{path}', Util.isArray(item.path) && item.path.join('.') || item.path)

.replace('{action}', item.action)

.replace('{expected}', item.expected)

.replace('{actual}', item.actual)

},

equal: function(type, path, actual, expected, result, message) {

if (actual === expected) return true

switch (type) {

case 'type':

// 正则模板 === 字符串最终值

if (expected === 'regexp' && actual === 'string') return true

break

}

var item = {

path: path,

type: type,

actual: actual,

expected: expected,

action: 'is equal to',

message: message

}

item.message = Assert.message(item)

result.push(item)

return false

},

// actual matches expected

match: function(type, path, actual, expected, result, message) {

if (expected.test(actual)) return true

var item = {

path: path,

type: type,

actual: actual,

expected: expected,

action: 'matches',

message: message

}

item.message = Assert.message(item)

result.push(item)

return false

},

notEqual: function(type, path, actual, expected, result, message) {

if (actual !== expected) return true

var item = {

path: path,

type: type,

actual: actual,

expected: expected,

action: 'is not equal to',

message: message

}

item.message = Assert.message(item)

result.push(item)

return false

},

greaterThan: function(type, path, actual, expected, result, message) {

if (actual > expected) return true

var item = {

path: path,

type: type,

actual: actual,

expected: expected,

action: 'is greater than',

message: message

}

item.message = Assert.message(item)

result.push(item)

return false

},

lessThan: function(type, path, actual, expected, result, message) {

if (actual < expected) return true

var item = {

path: path,

type: type,

actual: actual,

expected: expected,

action: 'is less to',

message: message

}

item.message = Assert.message(item)

result.push(item)

return false

},

greaterThanOrEqualTo: function(type, path, actual, expected, result, message) {

if (actual >= expected) return true

var item = {

path: path,

type: type,

actual: actual,

expected: expected,

action: 'is greater than or equal to',

message: message

}

item.message = Assert.message(item)

result.push(item)

return false

},

lessThanOrEqualTo: function(type, path, actual, expected, result, message) {

if (actual <= expected) return true

var item = {

path: path,

type: type,

actual: actual,

expected: expected,

action: 'is less than or equal to',

message: message

}

item.message = Assert.message(item)

result.push(item)

return false

}

}

valid.Diff = Diff

valid.Assert = Assert

module.exports = valid

var Util = require('../util')

// 备份原生 XMLHttpRequest

window.\_XMLHttpRequest = window.XMLHttpRequest

window.\_ActiveXObject = window.ActiveXObject

try {

new window.Event('custom')

} catch (exception) {

window.Event = function(type, bubbles, cancelable, detail) {

var event = document.createEvent('CustomEvent') // MUST be 'CustomEvent'

event.initCustomEvent(type, bubbles, cancelable, detail)

return event

}

}

var XHR\_STATES = {

// The object has been constructed.

UNSENT: 0,

// The open() method has been successfully invoked.

OPENED: 1,

// All redirects (if any) have been followed and all HTTP headers of the response have been received.

HEADERS\_RECEIVED: 2,

// The response's body is being received.

LOADING: 3,

// The data transfer has been completed or something went wrong during the transfer (e.g. infinite redirects).

DONE: 4

}

var XHR\_EVENTS = 'readystatechange loadstart progress abort error load timeout loadend'.split(' ')

var XHR\_REQUEST\_PROPERTIES = 'timeout withCredentials'.split(' ')

var XHR\_RESPONSE\_PROPERTIES = 'readyState responseURL status statusText responseType response responseText responseXML'.split(' ')

// https://github.com/trek/FakeXMLHttpRequest/blob/master/fake\_xml\_http\_request.js#L32

var HTTP\_STATUS\_CODES = {

100: "Continue",

101: "Switching Protocols",

200: "OK",

201: "Created",

202: "Accepted",

203: "Non-Authoritative Information",

204: "No Content",

205: "Reset Content",

206: "Partial Content",

300: "Multiple Choice",

301: "Moved Permanently",

302: "Found",

303: "See Other",

304: "Not Modified",

305: "Use Proxy",

307: "Temporary Redirect",

400: "Bad Request",

401: "Unauthorized",

402: "Payment Required",

403: "Forbidden",

404: "Not Found",

405: "Method Not Allowed",

406: "Not Acceptable",

407: "Proxy Authentication Required",

408: "Request Timeout",

409: "Conflict",

410: "Gone",

411: "Length Required",

412: "Precondition Failed",

413: "Request Entity Too Large",

414: "Request-URI Too Long",

415: "Unsupported Media Type",

416: "Requested Range Not Satisfiable",

417: "Expectation Failed",

422: "Unprocessable Entity",

500: "Internal Server Error",

501: "Not Implemented",

502: "Bad Gateway",

503: "Service Unavailable",

504: "Gateway Timeout",

505: "HTTP Version Not Supported"

}

/\*

MockXMLHttpRequest

\*/

function MockXMLHttpRequest() {

// 初始化 custom 对象，用于存储自定义属性

this.custom = {

events: {},

requestHeaders: {},

responseHeaders: {}

}

}

MockXMLHttpRequest.\_settings = {

timeout: '10-100',

/\*

timeout: 50,

timeout: '10-100',

\*/

}

MockXMLHttpRequest.setup = function(settings) {

Util.extend(MockXMLHttpRequest.\_settings, settings)

return MockXMLHttpRequest.\_settings

}

Util.extend(MockXMLHttpRequest, XHR\_STATES)

Util.extend(MockXMLHttpRequest.prototype, XHR\_STATES)

// 标记当前对象为 MockXMLHttpRequest

MockXMLHttpRequest.prototype.mock = true

// 是否拦截 Ajax 请求

MockXMLHttpRequest.prototype.match = false

// 初始化 Request 相关的属性和方法

Util.extend(MockXMLHttpRequest.prototype, {

// https://xhr.spec.whatwg.org/#the-open()-method

// Sets the request method, request URL, and synchronous flag.

open: function(method, url, async, username, password) {

var that = this

Util.extend(this.custom, {

method: method,

url: url,

async: typeof async === 'boolean' ? async : true,

username: username,

password: password,

options: {

url: url,

type: method

}

})

this.custom.timeout = function(timeout) {

if (typeof timeout === 'number') return timeout

if (typeof timeout === 'string' && !~timeout.indexOf('-')) return parseInt(timeout, 10)

if (typeof timeout === 'string' && ~timeout.indexOf('-')) {

var tmp = timeout.split('-')

var min = parseInt(tmp[0], 10)

var max = parseInt(tmp[1], 10)

return Math.round(Math.random() \* (max - min)) + min

}

}(MockXMLHttpRequest.\_settings.timeout)

// 查找与请求参数匹配的数据模板

var item = find(this.custom.options)

function handle(event) {

// 同步属性 NativeXMLHttpRequest => MockXMLHttpRequest

for (var i = 0; i < XHR\_RESPONSE\_PROPERTIES.length; i++) {

try {

that[XHR\_RESPONSE\_PROPERTIES[i]] = xhr[XHR\_RESPONSE\_PROPERTIES[i]]

} catch (e) {}

}

// 触发 MockXMLHttpRequest 上的同名事件

that.dispatchEvent(new Event(event.type /\*, false, false, that\*/ ))

}

// 如果未找到匹配的数据模板，则采用原生 XHR 发送请求。

if (!item) {

// 创建原生 XHR 对象，调用原生 open()，监听所有原生事件

var xhr = createNativeXMLHttpRequest()

this.custom.xhr = xhr

// 初始化所有事件，用于监听原生 XHR 对象的事件

for (var i = 0; i < XHR\_EVENTS.length; i++) {

xhr.addEventListener(XHR\_EVENTS[i], handle)

}

// xhr.open()

if (username) xhr.open(method, url, async, username, password)

else xhr.open(method, url, async)

// 同步属性 MockXMLHttpRequest => NativeXMLHttpRequest

for (var j = 0; j < XHR\_REQUEST\_PROPERTIES.length; j++) {

try {

xhr[XHR\_REQUEST\_PROPERTIES[j]] = that[XHR\_REQUEST\_PROPERTIES[j]]

} catch (e) {}

}

return

}

// 找到了匹配的数据模板，开始拦截 XHR 请求

this.match = true

this.custom.template = item

this.readyState = MockXMLHttpRequest.OPENED

this.dispatchEvent(new Event('readystatechange' /\*, false, false, this\*/ ))

},

// https://xhr.spec.whatwg.org/#the-setrequestheader()-method

// Combines a header in author request headers.

setRequestHeader: function(name, value) {

// 原生 XHR

if (!this.match) {

this.custom.xhr.setRequestHeader(name, value)

return

}

// 拦截 XHR

var requestHeaders = this.custom.requestHeaders

if (requestHeaders[name]) requestHeaders[name] += ',' + value

else requestHeaders[name] = value

},

timeout: 0,

withCredentials: false,

upload: {},

// https://xhr.spec.whatwg.org/#the-send()-method

// Initiates the request.

send: function send(data) {

var that = this

this.custom.options.body = data

// 原生 XHR

if (!this.match) {

this.custom.xhr.send(data)

return

}

// 拦截 XHR

// X-Requested-With header

this.setRequestHeader('X-Requested-With', 'MockXMLHttpRequest')

// loadstart The fetch initiates.

this.dispatchEvent(new Event('loadstart' /\*, false, false, this\*/ ))

if (this.custom.async) setTimeout(done, this.custom.timeout) // 异步

else done() // 同步

function done() {

that.readyState = MockXMLHttpRequest.HEADERS\_RECEIVED

that.dispatchEvent(new Event('readystatechange' /\*, false, false, that\*/ ))

that.readyState = MockXMLHttpRequest.LOADING

that.dispatchEvent(new Event('readystatechange' /\*, false, false, that\*/ ))

that.status = 200

that.statusText = HTTP\_STATUS\_CODES[200]

// fix #92 #93 by @qddegtya

that.response = that.responseText = JSON.stringify(

convert(that.custom.template, that.custom.options),

null, 4

)

that.readyState = MockXMLHttpRequest.DONE

that.dispatchEvent(new Event('readystatechange' /\*, false, false, that\*/ ))

that.dispatchEvent(new Event('load' /\*, false, false, that\*/ ));

that.dispatchEvent(new Event('loadend' /\*, false, false, that\*/ ));

}

},

// https://xhr.spec.whatwg.org/#the-abort()-method

// Cancels any network activity.

abort: function abort() {

// 原生 XHR

if (!this.match) {

this.custom.xhr.abort()

return

}

// 拦截 XHR

this.readyState = MockXMLHttpRequest.UNSENT

this.dispatchEvent(new Event('abort', false, false, this))

this.dispatchEvent(new Event('error', false, false, this))

}

})

// 初始化 Response 相关的属性和方法

Util.extend(MockXMLHttpRequest.prototype, {

responseURL: '',

status: MockXMLHttpRequest.UNSENT,

statusText: '',

// https://xhr.spec.whatwg.org/#the-getresponseheader()-method

getResponseHeader: function(name) {

// 原生 XHR

if (!this.match) {

return this.custom.xhr.getResponseHeader(name)

}

// 拦截 XHR

return this.custom.responseHeaders[name.toLowerCase()]

},

// https://xhr.spec.whatwg.org/#the-getallresponseheaders()-method

// http://www.utf8-chartable.de/

getAllResponseHeaders: function() {

// 原生 XHR

if (!this.match) {

return this.custom.xhr.getAllResponseHeaders()

}

// 拦截 XHR

var responseHeaders = this.custom.responseHeaders

var headers = ''

for (var h in responseHeaders) {

if (!responseHeaders.hasOwnProperty(h)) continue

headers += h + ': ' + responseHeaders[h] + '\r\n'

}

return headers

},

overrideMimeType: function( /\*mime\*/ ) {},

responseType: '', // '', 'text', 'arraybuffer', 'blob', 'document', 'json'

response: null,

responseText: '',

responseXML: null

})

// EventTarget

Util.extend(MockXMLHttpRequest.prototype, {

addEventListener: function addEventListener(type, handle) {

var events = this.custom.events

if (!events[type]) events[type] = []

events[type].push(handle)

},

removeEventListener: function removeEventListener(type, handle) {

var handles = this.custom.events[type] || []

for (var i = 0; i < handles.length; i++) {

if (handles[i] === handle) {

handles.splice(i--, 1)

}

}

},

dispatchEvent: function dispatchEvent(event) {

var handles = this.custom.events[event.type] || []

for (var i = 0; i < handles.length; i++) {

handles[i].call(this, event)

}

var ontype = 'on' + event.type

if (this[ontype]) this[ontype](event)

}

})

// Inspired by jQuery

function createNativeXMLHttpRequest() {

var isLocal = function() {

var rlocalProtocol = /^(?:about|app|app-storage|.+-extension|file|res|widget):$/

var rurl = /^([\w.+-]+:)(?:\/\/([^\/?#:]\*)(?::(\d+)|)|)/

var ajaxLocation = location.href

var ajaxLocParts = rurl.exec(ajaxLocation.toLowerCase()) || []

return rlocalProtocol.test(ajaxLocParts[1])

}()

return window.ActiveXObject ?

(!isLocal && createStandardXHR() || createActiveXHR()) : createStandardXHR()

function createStandardXHR() {

try {

return new window.\_XMLHttpRequest();

} catch (e) {}

}

function createActiveXHR() {

try {

return new window.\_ActiveXObject("Microsoft.XMLHTTP");

} catch (e) {}

}

}

// 查找与请求参数匹配的数据模板：URL，Type

function find(options) {

for (var sUrlType in MockXMLHttpRequest.Mock.\_mocked) {

var item = MockXMLHttpRequest.Mock.\_mocked[sUrlType]

if (

(!item.rurl || match(item.rurl, options.url)) &&

(!item.rtype || match(item.rtype, options.type.toLowerCase()))

) {

// console.log('[mock]', options.url, '>', item.rurl)

return item

}

}

function match(expected, actual) {

if (Util.type(expected) === 'string') {

return expected === actual

}

if (Util.type(expected) === 'regexp') {

return expected.test(actual)

}

}

}

// 数据模板 ＝> 响应数据

function convert(item, options) {

return Util.isFunction(item.template) ?

item.template(options) : MockXMLHttpRequest.Mock.mock(item.template)

}

module.exports = MockXMLHttpRequest