详情页组件制作（测试用例）

先定义组件文件，pages/detail/CaseView.vue

```
<template>
  <el-row justify="center">
    <el-col :span="24">
      <breadcrumb></breadcrumb>
    </el-col>
  </el-row>
  <el-row>
    <el-col :span="24">
      <div>
        详情页
      </div>
    </el-col>
  </el-row>
</template>

<script>
import Breadcrumb from '@/components/common/Breadcrumb.vue'

export default {
  components:{
    Breadcrumb,
  },
}
</script>
<style scoped>
.el-row {
  margin-bottom: 20px;
  &:last-child {
    margin-bottom: 0;
  }
}
.el-col {
  border-radius: 4px;
}
.grid-content {
  border-radius: 4px;
  min-height: 36px;
}
.row-bg {
  padding: 10px 0;
  background-color: #f9fafc;
}
</style>
```

为详情页设计入口，路由模式为`/cases/<int:caseid>`

添加路由 router/index.js

```
import { createRouter, createWebHistory } from 'vue-router'
import Home from '../views/Home.vue'
```

```
import Haiwen from '../views/Haiwen.vue'
import Afeng from '../views/Afeng.vue'
import Login from '../views/Login.vue'

const routes = [  //路由列表
  {
    path: '/',   // 请求的路径从Host之后开始计算--一级路由
    name: 'Home',  // 路由名称，方便后续引用
    meta:{
      title: '首页'
    },
    component: Home,  // 组件
    children:[  //子路由，这里的路由组件会显示到当前父组件的router-view中
      {
        path: 'cases', //二级路由这里开头不需要加/
        meta:{  //存放自定义数据
          title: '测试用例',
          icon: 'el-icon-s-order',
        },
        component: ()=>import("../pages/Cases.vue"),
        children:[
          {
            path: ':chapters(\\d+)+',
            component: ()=>import("../pages/detail/CaseView.vue"),
            meta:{
              title: '测试用例详情页'
            }
          }
        ]
      },{
        path: 'request',
        meta:{  //存放自定义数据
          title: 'web接口',
          icon: 'el-icon-s-promotion',
        },
        component: ()=>import("../pages/Request.vue")
      },{
        path: 'plans',
        meta:{  //存放自定义数据
          title: '测试计划',
          icon: 'el-icon-s-flag',
        },
        component: ()=>import("../pages/Plans.vue")
      },{
        path: 'reports',
        meta:{  //存放自定义数据
          title: '测试报告',
          icon: 'el-icon-s-data',
        },
        component: ()=>import("../pages/Reports.vue")
      }
    ]
  },
  {
    path: '/about',
    name: 'About',
    // route level code-splitting
    // this generates a separate chunk (about.[hash].js) for this route
```

```
    // which is lazy-loaded when the route is visited.
    component: () => import(/* webpackChunkName: "about" */
'../views/About.vue')
  },
  {
    path: '/haiwen',
    name: 'haiwen',
    component: Haiwen
  },
  {
    path: '/afeng',
    name: 'afeng',
    component: Afeng
  },
  {
    path: '/login',
    name: 'login',
    component: Login
  },
]
// 创建了router
const router = createRouter({
  history: createWebHistory(process.env.BASE_URL),
  routes
})

//全局路由前置守卫,
router.beforeEach((to,from,next)=>{ //回调接收to,from, next
  if(to.name!=='login' && localStorage.getItem('islogin')!=='yes'){  //如果不访问
login页面,且没有登录，就重定向到login
    next('/login')
  }else{
    next()
  }
})
export default router
```

表格组件这里将编辑按钮设计为用例的详情页入口

Tables.vue

```
<template>
  <el-table :data="tableData" style="width: 100%">
    <el-table-column type="selection" width="55"> </el-table-column>

    <el-table-column :label="item.title" width="180" :key="index" v-for="
(item,index) in columns">
      <template #default="scope">
        <i class="item.icon" v-if="item.icon"></i>
        <span style="margin-left: 10px">{{field_value(scope.row,item.field) }}
</span>
      </template>
    </el-table-column>

    <el-table-column label="操作">
      <template #default="scope">
```

```
        <el-button size="mini" @click="handleEdit(scope.$index, scope.row)"
          >编辑</el-button
        >
        <el-button
          size="mini"
          type="danger"
          @click="handleDelete(scope.$index, scope.row)"
          >删除</el-button
        >
      </template>
    </el-table-column>
  </el-table>
</template>

<script>
import { inject } from '@vue/runtime-core';
import { useRoute, useRouter } from 'vue-router';


export default {
  props:{
    columns: Array,
    tableData: Object
  },
  setup(){
    const columns = inject('columns')
    const tableData = inject('tableData')
    const router = useRouter()
    const route = useRoute()
    function field_value(obj,fields){
      //定义一个临时对象
      let temp_obj= obj
      //根据点来分割多级字段
      for(const item of fields.split('.')){
        //通过Reflect反复获取字段
        temp_obj=Reflect.get(temp_obj,item)
      }
      return temp_obj
    }

    function handleEdit(index, row) {
      router.push(`${route.path}/${row.id}`)
    }
    function handleDelete(index, row) {
      console.log(index, row);
    }
    return{
      handleEdit,
      handleDelete,
      field_value,
      columns,
      tableData
    }
  }
};
</script>

<style>
```

```
</style>
```

更新case路由组件显示逻辑,否则无法显示详情页内容

pages/Cases.vue

```
<template>
  <main-layout v-if="$route.path=='/cases'"></main-layout>
  <router-view v-else></router-view>
</template>
...
```

详情页表单制作-框架

```
<template>
  <el-row justify="center">
    <el-col :span="24">
      <breadcrumb></breadcrumb>
    </el-col>
  </el-row>
  <el-row>
    <el-col :span="24">
      <el-form
        label-position="left"
        label-width="80px"
        :model="caseForm"
      >
        <el-form-item>
          <div class="title"><strong>Common</strong></div>
        </el-form-item>
        <el-form-item label="file_path">
          <el-input v-model="caseForm.file_path"></el-input>
        </el-form-item>
        <el-form-item label="desc">
          <el-input v-model="caseForm.desc"></el-input>
        </el-form-item>
        <el-form-item label="project">
          <el-input v-model="caseForm.project"></el-input>
        </el-form-item>
        <el-form-item>
          <div class="title"><strong>Config</strong></div>
        </el-form-item>
        <el-form-item label="name">
          <el-input></el-input>
        </el-form-item>
        <el-form-item label="base_url">
          <el-input></el-input>
        </el-form-item>
        <el-form-item label="variables">
          <el-input  type="textarea"></el-input>
        </el-form-item>
        <el-form-item label="parameters">
          <el-input  type="textarea"></el-input>
```

```
        </el-form-item>
        <el-form-item label="verify">
          <el-switch v-model="caseForm.status"></el-switch>
        </el-form-item>
        <el-form-item label="export">
          <el-input  type="textarea"></el-input>
        </el-form-item>
        <el-form-item>
          <div class="title">
            <strong>TestSteps</strong>
            <button class="newLine">添加一行</button>
          </div>
        </el-form-item>
        <el-form-item v-for="(item,index) in caseForm.steps" :label="item.no">
          <el-input type="textarea" v-model="item.stepContent"></el-input>
        </el-form-item>

      </el-form>
    </el-col>
  </el-row>
</template>

<script>
import Breadcrumb from "@/components/common/Breadcrumb.vue";
import { reactive, ref } from '@vue/reactivity';

export default {
  components: {
    Breadcrumb,
  },
  setup(){
    const caseForm = reactive({
      file_path:'file_path',
      desc:'desc',
      project:'',
      status: false,
      steps:[
        {
          no: 1,
          stepContent:123
        },
        {
          no: 2,
          stepContent:666
        }
      ]
    })
    return{
      caseForm
    }
  }
};
</script>
<style scoped>
.el-row {
  margin-bottom: 20px;
  &:last-child {
    margin-bottom: 0;
```

```
  }
}
.el-col {
  border-radius: 4px;
}
.grid-content {
  border-radius: 4px;
  min-height: 36px;
}
.row-bg {
  padding: 10px 0;
  background-color: #f9fafc;
}
div.title{
  text-align: left;
}
.newLine{
  float: right;
}
</style>
```

详情页表单制作-数据回显

数据回显原理是像后端请求数据，然后再渲染到前端页面上。

有了vue的响应式，实现这个过程可以很简单。

新增用例详情获取方法

```
//用例详情
function caseDetail(case_id){
  return axios({
    method: 'get',
    url:`/api/cases/${case_id}`,
  })
}
```

组件中使用

```
const route = useRoute()
const case_id = route.path.split('/').pop()
caseDetail(case_id).then((res)=>{
console.log(res.data);
caseForm = res.data
})
```


前端技术总结(vue部分)

vue开发模式-组件化开发

含义：页码分割成大大小小的模块，入导航栏，菜单项，表格，分页等等，拼页面就像拼积木一样，实现模块化开发。

组件定义

通过创建Vue文件，包含template script(可选) style（可选）

组件的使用

1.引入组件文件

2.注册组件

3.模板中使用组件名作为标签

组件的通信

父子通信： props

跨组件通信: provide/inject

插槽的使用

作用：使得组件可以在指定位置显示其他组件的HTML内容

普通插槽`<slot></slot>`

具名插槽`<slot name="demo"></slot>`