顶栏的退出按钮实现点击退出效果

菜单项动态渲染

路由信息配置,原路由信息加上meta配置

```
const routes = [  //路由列表
  {
    path: '/',   // 请求的路径从Host之后开始计算
    name: 'Home',  // 路由名称，方便后续引用
    component: Home,  // 组件
    children:[{
      path: 'cases',
      component: () => import('../pages/Cases.vue'),
      meta:{
        icon: 'el-icon-s-order',
        title: '测试用例'
      }
    },{
      path: 'request',
      component: () => import('../pages/Request.vue'),
      meta:{
        icon: 'el-icon-s-promotion',
        title: 'web接口'
      }
    },{
      path: 'plans',
      component: () => import('../pages/Plans.vue'),
      meta:{
        icon: 'el-icon-s-flag',
        title: '测试计划'
      }
    },{
      path: 'reports',
      component: () => import('../pages/Reports.vue'),
      meta:{
        icon: 'el-icon-s-data',
        title: '测试报告'
      }
    }]
  },
  ...
  ]
```

自定义菜单栏组件components/common/sidebaritem.vue

```
<template>
  <el-menu-item :index="route.path" >
    <i :class="route.meta.icon"></i>
    <template #title>{{route.meta.title}}</template>
  </el-menu-item>
</template>
```

```
<script>
export default {
  name:'SidebarItem',
  props: {
    route: Object
  },
  setup(props) {

    return{
    }
  }
}
</script>
```

替换侧边栏-Sidbar.vue

```
<template>
    <el-menu
      router
      :uniqueOpened="true"
      default-active="2"
      class="siderbar"
      background-color="#545c64"
      text-color="#fff"
      active-text-color="#ffd04b"
      :collapse="iscollapse"
    >
      <sidebar-item v-for="item in routes[0].children" :route="item"></sidebar-
item>

      <el-menu-item @click="switch_sidebar" class="switch_bar">
        <i class="el-icon-arrow-left" v-if="iscollapse===false"></i>
        <i class="el-icon-arrow-right" v-else></i>
      </el-menu-item>
      <!-- <SidebarItem :path="api" :route="abc" :icon="el-icon-s-order"
:title="test_menu"></SidebarItem> -->
    </el-menu>
</template>

<script>
  import {ref} from 'vue'
  import {useRouter} from 'vue-router'
  import SidebarItem from './common/SidebarItem.vue'
  export default {
    components:{
      SidebarItem
    },
    setup(){
      const router = useRouter()
      const routes = router.options.routes
      const iscollapse = ref(false)
      function switch_sidebar(){
        iscollapse.value=!iscollapse.value
      }

      return{
```

```
        iscollapse,
        switch_sidebar,
        routes
      }
    }
  }
</script>

<style>
.siderbar{
  /* 撑开测边栏 */
  height: 100vh;
}
</style>
```

主区域布局

主区域采用面包屑加图表布局的方法。这里可以参考element-plus现成的布局元素,稍许改动下

components/common/MainLayout.vue

```
<template>
<el-row justify="center" >
  <el-col :span="22"><div class="grid-content bg-purple-dark">面包屑</div></el-
col>
</el-row>
<el-row justify="center" align="bottom">
  <el-col :span="22"><div class="grid-content bg-purple">图表</div></el-col>
</el-row>
</template>

<style>
  .el-row {
    margin-bottom: 20px;
    &:last-child {
      margin-bottom: 0;
    }
  }
  .el-col {
    border-radius: 4px;
  }
  .bg-purple-dark {
    background: #99a9bf;
  }
  .bg-purple {
    background: #d3dce6;
  }
  .bg-purple-light {
    background: #e5e9f2;
  }
  .grid-content {
    border-radius: 4px;
    min-height: 36px;
  }
```

```
  .row-bg {
    padding: 10px 0;
    background-color: #f9fafc;
  }
</style>
```

将组件导入到cases.vue中,访问/cases进行调试

```
<template>
  <main-layout></main-layout>
</template>

<script>
import MainLayout from '../components/common/MainLayout.vue'
export default {
  components:{
    MainLayout
  }
}
</script>
```

面包屑设计

现用现成的组件测试一下效果

components/common/Breadcrumb.vue

```
<template>
  <el-breadcrumb separator="/">
  <el-breadcrumb-item :to="{ path: '/' }">首页</el-breadcrumb-item>
  <el-breadcrumb-item><a href="/">活动管理</a></el-breadcrumb-item>
  <el-breadcrumb-item>活动列表</el-breadcrumb-item>
  <el-breadcrumb-item>活动详情</el-breadcrumb-item>
</el-breadcrumb>
</template>
```

嵌套进布局中查看效果

MainLayout.vue

```
<template>
<el-row justify="center" align="middle">
  <el-col :span="22">
    <breadcrumb></breadcrumb>
  </el-col>
</el-row>
<el-row justify="center" align="bottom">
  <el-col :span="22"><div class="grid-content bg-purple">图表</div></el-col>
</el-row>
</template>

<script>
import Breadcrumb from './Breadcrumb.vue'
export default {
  components:{
    Breadcrumb
```

```
  }
}
</script>
```

同样，面包屑这里希望能够自动生成菜单，所以做成一个通用组件。

```
<template>
<el-breadcrumb separator="/">
  <!-- <el-breadcrumb-item :to="{ path: '/' }">首页</el-breadcrumb-item> -->
  <el-breadcrumb-item v-for="(item,index) in currrentRouteList">
    <a :href="item.path">{{item.name}}</a>
  </el-breadcrumb-item>
</el-breadcrumb>
</template>

<script>
import { getCurrentInstance,  onMounted, reactive, watch } from 'vue'
import { useRoute } from 'vue-router'
export default {
  name: 'Breadcrumb',

  setup(props){
    const route = useRoute()
    const currrentRouteList= route.matched
    console.log(currrentRouteList);
    return{
      currrentRouteList
    }
  }

}
</script>
```

这里 的关键点是 `route.matched` 可以获取到当前页面匹配的路由和父路由

表格组件开发

数据的展示主要依靠表格部分，同样，我们开发出一套之后，其他的页面就可以复用这个页面，因为格式模板都是相同的，只是数据不同。

表格这里同样采用element-plus的自定义列模板和多选进行一下结合,形成最初的模板

components/common/Tables.vue

```
<template>
  <el-table :data="tableData" stripe style="width: 100%">
    <el-table-column type="selection" width="55"> </el-table-column>
    <el-table-column label="日期" width="180">
      <template #default="scope">
        <i class="el-icon-time"></i>
        <span style="margin-left: 10px">{{ scope.row.date }}</span>
      </template>
    </el-table-column>
```

```
    <el-table-column label="姓名" width="180">
      <template #default="scope">
        <el-popover effect="light" trigger="hover" placement="top">
          <template #default>
            <p>姓名: {{ scope.row.name }}</p>
            <p>住址: {{ scope.row.address }}</p>
          </template>
          <template #reference>
            <div class="name-wrapper">
              <el-tag size="medium">{{ scope.row.name }}</el-tag>
            </div>
          </template>
        </el-popover>
      </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>
export default {
  data() {
    return {
      tableData: [
        {
          date: "2021-09-16",
          name: "海文",
          address: "南京市软件大道23号 京妆商务505",
        },
        {
          date: "2016-05-04",
          name: "心田",
          address: "南京市软件大道23号 京妆商务505",
        },
        {
          date: "2016-05-01",
          name: "小猪",
          address: "南京市软件大道23号 京妆商务505",
        },
        {
          date: "2016-05-03",
          name: "苏三",
          address: "南京市软件大道23号 京妆商务505",
        },
      ],
    };
```

```
  },
  methods: {
    handleEdit(index, row) {
      console.log(index, row);
    },
    handleDelete(index, row) {
      console.log(index, row);
    },
  },
};
</script>

<style>
</style>
```

组装到MainLayout.vue看下效果

```
<template>
<el-row justify="center" align="middle">
  <el-col :span="22">
    <breadcrumb></breadcrumb>
  </el-col>
</el-row>
<el-row justify="center" align="bottom">
  <el-col :span="22">
    <tables></tables>
  </el-col>
</el-row>
</template>

<script>
import Breadcrumb from './Breadcrumb.vue'
import Tables from './Tables.vue'
export default {
  components:{
    Breadcrumb,
    Tables
  }
}
</script>
<style>
  .el-row {
    margin-bottom: 20px;
    &:last-child {
      margin-bottom: 0;
    }
  }
  .el-col {
    border-radius: 4px;
  }

</style>
```

寻思一下，table这里是否页可以做成一个通用的组件。这里依然采用动态渲染的思路来解决问题。

```
<template>
  <el-table :data="tableData" stripe >
    <el-table-column type="selection" width="55"></el-table-column>
    <el-table-column v-for="(item,index) in columns" :label="item.title"
width="180">
      <template #default="scope">
        <i :class="item.icon" v-if="item.icon"></i>
        <span style="margin-left: 10px">{{ 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 { reactive } from 'vue';
export default {
  setup() {
    const columns=[
      {
        title: '日期',
        field: 'date',
        icon: 'el-icon-time'
      },{
        title: '姓名',
        field: 'name',
      },{
        title: '地址',
        field: 'address',
      }
    ]
    const tableData= reactive([
        {
          date: "2021-09-16",
          name: "海文",
          address: "南京市软件大道23号 京妆商务505",
        },
        {
          date: "2016-05-04",
          name: "心田",
          address: "南京市软件大道23号 京妆商务505",
        },
        {
          date: "2016-05-01",
```

```
            name: "小猪",
            address: "南京市软件大道23号 京妆商务505",
          },
          {
            date: "2016-05-03",
            name: "苏三",
            address: "南京市软件大道23号 京妆商务505",
          },
        ])
      function handleEdit(index, row) {
        console.log(index, row);
      }
      function handleDelete(index, row) {
        console.log(index, row);
      }
      return {
        tableData,
        handleEdit,
        handleDelete,
        columns
      };
    }
}
</script>

<style>
</style>
```

将main_layout放到其他pages组件中进行调试，查看效果。

前后端数据交互

目前数据都是写死的，所以大家在每个页面上看的都一样，为了模拟真实的效果，现在开放出了数据交互接口。

```
//查询用例请求
http://120.27.146.185:8076/api/cases/?page_size=5&page_index=2
//查询web接口
http://120.27.146.185:8076/api/requests/?page_size=5&page_index=1
//查询测试计划
http://120.27.146.185:8076/api/plans/?page_size=5&page_index=1
//查询测试报告
http://120.27.146.185:8076/api/reports/?page_size=5&page_index=1
```

httplib/index.js增加请求方法

```
function getCases(page_size,page_index){
  return axios({
    method: 'get',
    url: 'api/cases/',
    params: {
      page_size,
      page_index,
    }
  })
}
```

MainLayout.vue

```
<template>
<el-row justify="center" align="middle">
  <el-col :span="22">
    <breadcrumb></breadcrumb>
  </el-col>
</el-row>
<el-row justify="center" align="bottom">
  <el-col :span="22">
    <tables :columns="columns" :tableData="tableData"></tables>
  </el-col>
</el-row>
</template>

<script>
import {reactive, ref} from 'vue'
import Breadcrumb from './Breadcrumb.vue'
import Tables from './Tables.vue'
import {getCases} from '@/httplib'
export default {
  components:{
    Breadcrumb,
    Tables
  },
  setup(){
    const columns=[
      {
        title: '用例名称',
        field: 'config.name',
      },{
        title: '所属项目',
        field: 'config.project.name',
      },{
        title: '文件路径',
        field: 'file_path',
      },{
        title: '创建时间',
        field: 'create_time',
        icon: 'el-icon-time'
      },{
        title: '更新时间',
        field: 'update_time',
        icon: 'el-icon-time'
      }
```

```
    ]
    const tableData= ref([])
    getCases(5,1).then(
      function(resp){
        tableData.value = resp.data.retlist
      }
    )

    return {
      columns,
      tableData
    }
  }
}
</script>
<style>
  .el-row {
    margin-bottom: 20px;
    &:last-child {
      margin-bottom: 0;
    }
  }
  .el-col {
    border-radius: 4px;
  }

</style>
```

Tables.vue

```
<template>
  <el-table :data="tableData" stripe >
    <el-table-column type="selection" width="55"></el-table-column>
    <el-table-column v-for="(item,index) in columns" :label="item.title"
width="180">
      <template #default="scope">
        <i :class="item.icon" v-if="item.icon"></i>
        <span style="margin-left: 10px">{{ 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 { reactive } from 'vue';
export default {
  props:{
    columns:Array,
    tableData: Object
  },
  setup() {

    function handleEdit(index, row) {
      console.log(index, row);
    }
    function handleDelete(index, row) {
      console.log(index, row);
    }
    return {
      handleEdit,
      handleDelete,
    };
  }
}
</script>

<style>
</style>
```