Vue2Project DAY01

基于嵌套路由实现main部分内容的动态更新

```
http://localhost:8080/component/container
http://localhost:8080/component/form
http://localhost:8080/component/table
```

设计嵌套路由：

```
当访问：http://localhost:8080/component/container
      看到Component组件中嵌套Container组件
http://localhost:8080/component/form
      看到Component组件中嵌套Form组件
http://localhost:8080/component/table
      看到Component组件中嵌套Table组件
```

实现步骤：

1. 先创建3个子组件：Container.vue Table.vue Form.vue
2. 在router/index.js中配置嵌套路由（为了实现上述目标进行配置。配置children）
3. 在el-main里添加一个二级路由占位符：`<router-view/>`
4. 测试：

   ```
   http://localhost:8080/component/container
   http://localhost:8080/component/form
   http://localhost:8080/component/table
   ```

动态加载左侧边栏菜单列表

未来不同的用户有可能看到的侧边栏列表也不同，所以左侧边栏的内容极有可能是发请求从后端动态加载，通过v-for遍历显示的。例如，发送一个请求，获取当前用户可以看到的侧边栏菜单信息，得到如下json：

```
[
    {
        path: '/component/container',
        icon: 'el-icon-menu',
        text: 'Container组件'
    },
    {
        path: '/component/table',
        icon: 'el-icon-s-data',
        text: 'table组件'
    },
```

```
    {
        path: '/component/form',
        icon: 'el-icon-edit-outline',
        text: 'form组件'
    },
]
```

通过v-for即可动态加载。值得一提的是：**不同的用户返回的json有可能不同。**

如果后端返回的数据包含子菜单项，如下json：

```
[
    {
        path: '/component/index',
        icon: 'el-icon-s-home',
        text: '首页'
    },
    {
        path: '/components',
        icon: 'el-icon-menu',
        text: '常用组件',
        children: [
            {
                path: '/component/container',
                icon: 'el-icon-menu',
                text: 'Container组件'
            },
            {
                path: '/component/table',
                icon: 'el-icon-s-data',
                text: 'table组件'
            },
            {
                path: '/component/form',
                icon: 'el-icon-edit-outline',
                text: 'form组件'
            },
        ]
    },
]
```

```
[
    {text:'在吗', user:'you', id:1},
    {text:'在', user:'me', id:2},
    {text:'怎么了', user:'me', id:3},
    {text:'中午吃饭？', user:'you', id:4},
    {text:'你请客？', user:'me', id:5},
    {text:'没问题', user:'you', id:6},
]
```

Table组件

Table组件用于显示表格数据。基本结构：

```
<el-table :data="tableData">
    <el-table-column prop="date" label="日期"> </el-table-column>
    <el-table-column prop="name" label="姓名"> </el-table-column>
    <el-table-column prop="address" label="地址"> </el-table-column>
</el-table>
```

为el-table提供:data属性，用于指定数据集。数据集的结构需要满足：

```
[{}, {}, {}, {}, {}, {}..]
```

el-table-column用于描述表格列，label属性为列名，prop属性为列中应该显示的对象的属性名，width对应列的宽度，align对齐方式。

表格数据显示中的自定义列模板

```
<el-table :data="tableData" style="width: 100%">
    <el-table-column label="日期" width="180">
        <template slot-scope="scope">
            <span style="margin-left: 10px">{{ scope.row.date }}</span>
        </template>
    </el-table-column>
</el-table>
```

Form表单组件

```
<el-form :model="form" label-width="80px">
    <el-form-item label="标签名">
        <el-input v-model="form.name"></el-input>
    </el-form-item>
    ....
</el-form>
```

```
data(){
    return {
        form: {
            name: ''
        }
    }
}
```

案例：

访问地址： /component/form， 提供注册页面。账号、密码、确认密码、手机号，提供提交按钮。

表单验证

无论使用任何技术进行验证，都需要完成以下两部分：

1. 当某一个表单项填写完毕后，进行验证。
2. 当点击提交按钮时，先做完整的表单验证，验证通过后再提交。

当某一个表单项填写完毕后，进行验证

```
<el-form :model="form"
         :rules="rules"
         label-width="80px" style="width:600px">
    <el-form-item label="账号" prop="name">
        <el-input v-model="form.name"></el-input>
    </el-form-item>
</el-form>
```

```
rules: {    // 定义验证规则
    name: [
        {required:true, message:'该字段必填', trigger:"blur"},
        {pattern:/^\d{6}$/, message:'必须6位数字', trigger:"blur"},
        {
            validator: (rule, value, callback)=>{
                if(value == this.form.pwd){
                    callback()
                }else {
                    callback(new Error('两次密码输入不一致'))
                }
            }, trigger: "blur"
        }
    ]
}
```

当点击提交按钮时，先做完整的表单验证，验证通过后再提交

知识点：通过this.$refs获取组件对象:

```
<el-form ref="form">
    <el-form-item>
        <el-input ref="input"></el-input>
    </el-form-item>
</el-form>
```

```
this.$refs.form       获取elForm表单组件对象
this.$refs['form']    获取elForm表单组件对象

this.$refs.input   获取elInput输入框组件对象
```

```
this.$refs.form.validate((valid)=>{
    if(valid){
        验证通过
    }else {
        验证失败
    }
})
```

1. 下载navicat：

   navicat下载链接：

   https://pan.baidu.com/s/1ObI8Ra9a3-gFot5snG6Q7A&pwd=1234

   解压缩，运行navicat.exe即可打开软件。

   创建一个mysql连接，可以正常连接mysql。

2. 运行sql脚本： 解压缩两个压缩包，得到两个mysql脚本文件：

   init_1.sql init2.sql

   右键navicat已经建立好的连接，选择运行sql脚本文件，依次选择init_1.sql init2.sql。

3. 在运行init2.sql时，有可能运行报错。

   原因是数据集太大了，一条sql语句的大小超出了xampp自带的mysql的限制，所以需要调整mysql的配置文件：my.ini，修改一些参数，才可以正常执行脚本：