解决多级字段不能显示的问题

问题：

测试用例菜单对应的表格中，用例名称和所属项目无法显示

原因是，这两个配置字段用的是多级字段

```
{
    title: '用例名称',  //列的标题
    field: 'config.name',       //数据的字段名
},{
    title: '所属项目',
    field: 'config.project.name',
}
```

而Tables.vue组件中没有进行处理

```
<span style="margin-left: 10px">{{ scope.row[item.field] }}</span>
```

`scope.row[item.field]`是没有办法处理多级字段的。

解决：

所以需要定义方法来获取多级字段的信息

```
function field_value(obj,fields){
    //操作零时对象
    let temp_obj = obj
    //根据点来分割多级字段
    for(const item of fields.split('.')){
    //通过反射来重复获取字段
    temp_obj = Reflect.get(temp_obj,item)
    }
    return temp_obj
}
```

替换Tables.vue中的取值部分

```
    <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>
                                        <!-- 这里替换掉 -->
        <span style="margin-left: 10px">{{ field_value(scope.row,item.field) }}
</span>
      </template>
    </el-table-column>
```

前后端交互续

继续完成web接口、测试计划、测试报告的数据展示。

httplib增加数据请求方法

htttplib/index.js

```
import axios from 'axios'  //引入ajax库-axios
import router from '../router'
import { ElMessage } from 'element-plus'  //导入消息框

axios.defaults.validateStatus=(status)=>{
  return status >=200 && status < 400 // 设置200-399之间的响应码为正常
}
axios.defaults.baseURL = 'http://120.27.146.185:8076'; //设置Host


//登录
function login(username,password){
  // 基本使用方法axios(config) config参考https://axios-http.com/zh/docs/req_config
  axios({
    method: 'post',
    url: '/api/login/',
    data:{
      username,
      password
    }
  }).then(
    //处理正常响应
    function(response){
      //成功后跳转到首页
      // console.log(response.data)
      ElMessage.success(response.data.msg)
      router.push('/')
      localStorage.setItem('islogin','yes')  //浏览器设置登录状态
    }
  ).catch(
    //处理异常响应，根据状态码，默认情况非2开头的响应码会在这里处理
    function(error){
      // console.log(error.response)
      ElMessage.error(error.response.data.error)
    }
  )
}

// 登出
function logout(){
  axios({
    method: 'get',
    url: '/api/logout/'
  }).then(()=>{
    //返回到登录
    router.push('/login')
    ElMessage.info('退出登录')
    localStorage.setItem('islogin','no')
  })
}

function common_get(url, page_size,page_index){
  return axios({
```

```
    method: 'get',
    url: url,
    params: {
      page_size,
      page_index
    }
  })
}
//获取用例数据
function getCases(page_size=5,page_index=1){
  return common_get('/api/cases/',page_size,page_index)
}
//获取web接口数据
function getRequest(page_size=5,page_index=1){
  return common_get('/api/requests/',page_size,page_index)
}
//获取测试计划数据
function getPlans(page_size=5,page_index=1){
  return common_get('/api/plans/',page_size,page_index)
}
//获取报告数据
function getReports(page_size=5,page_index=1){
  return common_get('/api/reports/',page_size,page_index)
}
//获取请求
export {login,logout,getCases,getRequest,getPlans,getReports}
```

Pages/Plans.vue

```
<template>
  <main-layout :tableData="tableData" :columns="columns"></main-layout>
</template>

<script>
import MainLayout from '../components/common/MainLayout.vue'
import {getPlans} from '@/httplib'
import { ref } from 'vue';

export default {
  components:{
    MainLayout
  },
    setup(){
     // 表格展示的字段信息
    const columns =[
      {
        title: '计划名称',  //列的标题
        field: 'name',      //数据的字段名
      },{
        title: '项目',
        field: 'environment.project.name',
      },{
        title: '测试人员',
        field: 'executor.username',
      },{
```

```
        title: '测试环境',
        field: 'environment.desc',
      },{
        title: '状态',
        field: 'status',
      },{
        title: '执行次数',
        field: 'exec_counts',
      },{
        title: '描述',
        field: 'desc',
      }
    ]
    const tableData = ref([])
    //读取后台请求:
    getPlans().then(
       function(resp){
         tableData.value = resp.data.retlist  //拿到后台返回的retlist
       }
     )
     return {
       columns,
       tableData,
     }
  }
}
</script>
```

Pages/Reports.vue

```
<template>
  <main-layout :tableData="tableData" :columns="columns"></main-layout>
</template>

<script>
import MainLayout from '../components/common/MainLayout.vue'
import {getReports} from '@/httplib'
import { ref } from 'vue';

export default {
  components:{
    MainLayout
  },
    setup(){
     // 表格展示的字段信息
    const columns =[
      {
        title: '测试计划',  //列的标题
        field: 'plan.name',       //数据的字段名
      },{
        title: '测试人员',
        field: 'trigger.username',
      },{
        title: '开始时间',
        field: 'create_time',
      },{
        title: '结束时间',
```

```
        field: 'update_time',
      },{
        title: '报告详情',
        field: 'desc',
      }
    ]
    const tableData = ref([])
    //读取后台请求:
    getReports().then(
       function(resp){
         tableData.value = resp.data.retlist  //拿到后台返回的retlist
       }
     )
     return {
       columns,
       tableData,
     }
  }
}
</script>
```

Pages/Request.vue

```
<template>
  <main-layout :tableData="tableData" :columns="columns"></main-layout>
</template>

<script>
import MainLayout from '../components/common/MainLayout.vue'
import {getRequest} from '@/httplib'
import { ref } from 'vue';

export default {
  components:{
    MainLayout
  },
    setup(){
     // 表格展示的字段信息
    const columns =[
      {
        title: '请求方法',  //列的标题
        field: 'method',      //数据的字段名
      },{
        title: '请求路径',
        field: 'url',
      },{
        title: '请求体参数',
        field: 'data',
      },{
        title: 'URL参数',
        field: 'params',
      },{
        title: '请求头',
        field: 'headers',
      }
    ]
    const tableData = ref([])
```

```
    //读取后台请求:
    getRequest().then(
       function(resp){
         tableData.value = resp.data.retlist  //拿到后台返回的retlist
       }
     )
     return {
       columns,
       tableData,
     }
  }
}
</script>
```

分页组件开发

添加分页组件

Pagination 分页 | ElementPlus (gitee.io)

components/common/paginator.vue

```
<template>
  <el-pagination background layout="prev, pager, next" :total="1000">
  </el-pagination>
</template>
```

使用分页组件

components/common/la.vue

```
<template>
  <el-row justify="center">
    <el-col :span="24">
      <breadcrumb></breadcrumb>
    </el-col>
  </el-row>
  <el-row>
    <el-col >
      <tables :tableData="tableData" :columns="columns"></tables>
    </el-col>
    <el-col>
      <paginator></paginator>
    </el-col>
  </el-row>
</template>

<script>
import Breadcrumb from './Breadcrumb.vue'
import Tables from './Tables.vue'
import Paginator from './paginator.vue'

export default {
  components:{
    Breadcrumb,
```

```
    Tables,
    Paginator
  },
  props:{
    columns: Array,
    tableData: Object
  },

}
</script>
```

测试分页组件的total（总数据条数）和page_size（当前页显示条数）

```
<template>
  <el-pagination background layout="prev, pager, next" :total="100" :page-
size="5">
  </el-pagination>
</template>
```

结合接口信息，我们知道page-size是前端传给后端的，total是后端返回的数据

```
<template>
  <el-pagination background layout="prev, pager, next" :total="total" :page-
size="page_size">
  </el-pagination>
</template>

<script>
export default {
  props:{
    total:Number,
    page_size:Number
  },
  setup() {

  },
}
</script>
```

作为入参接收total和page_size,父组件这里也引入传递

common/main_layout.vue

```
<template>
  <el-row justify="center">
    <el-col :span="24">
      <breadcrumb></breadcrumb>
    </el-col>
  </el-row>
  <el-row>
    <el-col >
      <tables :tableData="tableData" :columns="columns"></tables>
    </el-col>
    <el-col>
      <paginator :total="total" :page_size="page_size"></paginator>
    </el-col>
```

```
  </el-row>
</template>

<script>
import Breadcrumb from './Breadcrumb.vue'
import Tables from './Tables.vue'
import Paginator from './paginator.vue'

export default {
  components:{
    Breadcrumb,
    Tables,
    Paginator,
  },
  props:{
    columns: Array,
    tableData: Object,
    total:Number,
    page_size:Number
  },

}
</script>

<style scoped>
.el-row {
  margin-bottom: 20px;
  &:last-child {
    margin-bottom: 0;
  }
}
.el-col {
  border-radius: 4px;
}

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


分页组件与表格数据结合

分页组件要能正确控制表格中的数据，按照分页当前页显示数据。

通过自带的属性和事件可以控制当前页码和每页显示的条数

page-size: 控制每页显示的条数

currentPage： 控制页码

size-change 每页显示的条数改动时被触发

current-change 当前页码改动时被触发

```
<template>
  <el-pagination
    background
    layout="sizes,prev, pager, next"
    :total="total"
    :page-sizes="[5, 10, 20]"
    :page-size="page_size"
    @size-change="handleSizeChange"
    v-model:currentPage="page_index"
    @current-change="handleCurrentChange"
  >
  </el-pagination>
</template>

<script>
import { ref } from '@vue/reactivity';
export default {
  props:{
    total:Number,
    callback:Function
  },
  setup(props) {
    const page_size = ref(5)
    const page_index = ref(1)

    function handleSizeChange(size){
      console.log('page_size',size);
      page_size.value=size
      props.callback(page_size.value,page_index.value)
    }

    function handleCurrentChange(index){
      console.log('current page', index);
      page_index.value=index
      props.callback(page_size.value,page_index.value)
    }
    return {
      page_size,
      page_index,
      handleSizeChange,
      handleCurrentChange
    }
  },
};
</script>
```

表格数据从cases组件这里传递，所以我们把更新数据的方法丢给底层组件

Cases.vue

```
<template>
  <main-layout
    :tableData="tableData"
    :columns="columns"
    :total="total"
    :callback="sync_data"
  ></main-layout>
```

```
</template>

<script>
import MainLayout from "../components/common/MainLayout.vue";
import { getCases } from "@/httplib";
import { onMounted, ref } from "vue";

export default {
  components: {
    MainLayout,
  },
  setup() {
    // 表格展示的字段信息
    const columns = [
      {
        title: "用例名称", //列的标题
        field: "config.name", //数据的字段名
      },
      {
        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", //图标信息-非必填
      },
      {
        title: "更新时间",
        field: "update_time",
        icon: "el-icon-time", //图标信息-非必填
      },
    ];
    const tableData = ref([]);
    const total = ref(0);
    //读取后台请求：
    function sync_data(page_size=5,page_index=1){
      getCases(page_size,page_index).then(function (resp) {
        tableData.value = resp.data.retlist; //拿到后台返回的retlist
        total.value = resp.data.total;
      });
    }
    onMounted(()=>{
      sync_data()
    })
    return {
      columns,
      tableData,
      total,
      sync_data
    };
  },
};
</script>
```

组件的关系是Cases.vue>MainLayout.vue>paginator.vue

所以还要传给MainLayout.vue

```
<template>
  <el-row justify="center">
    <el-col :span="24">
      <breadcrumb></breadcrumb>
    </el-col>
  </el-row>
  <el-row>
    <el-col >
      <tables :tableData="tableData" :columns="columns"></tables>
    </el-col>
    <el-col>
      <paginator :total="total" :callback="callback"></paginator>
    </el-col>
  </el-row>
</template>

<script>
import Breadcrumb from './Breadcrumb.vue'
import Tables from './Tables.vue'
import Paginator from './paginator.vue'

export default {
  components:{
    Breadcrumb,
    Tables,
    Paginator,
  },
  props:{
    columns: Array,
    tableData: Object,
    total:Number,
    callback:Function
  },

}
</script>

<style scoped>
.el-row {
  margin-bottom: 20px;
  &:last-child {
    margin-bottom: 0;
  }
}
.el-col {
  border-radius: 4px;
}

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

```
</style>
```

跨组件通信

目前我们组件之间的数据通信还停留在父传子，子传孙的情况，但是如果组件的层级过深，这种方式就不合适了。因此vue推出了另种数据传递方式，Provide/Inject， 具体参考

Provide / Inject | Vue.js (vuejs.org)

Provide / Inject | Vue.js (vuejs.org) 组合式API的用法

总结一下就是父组件Provide数据，子组件Inject数据

根据以上规则，修改组件的数据传递方法

修改Case.vue的传递方法，改为Provide/Inject模式

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

<script>
import MainLayout from "../components/common/MainLayout.vue";
import { getCases } from "@/httplib";
import { onMounted, provide, ref } from "vue";

export default {
  components: {
    MainLayout,
  },
  setup() {
    // 表格展示的字段信息
    const columns = [
      {
        title: "用例名称", //列的标题
        field: "config.name", //数据的字段名
      },
      {
        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", //图标信息-非必填
      },
      {
        title: "更新时间",
        field: "update_time",
        icon: "el-icon-time", //图标信息-非必填
      },
    ];
    const tableData = ref([]);
    const total = ref(0);
```

```
    //读取后台请求:
    function sync_data(page_size=5,page_index=1){
      getCases(page_size,page_index).then(function (resp) {
        tableData.value = resp.data.retlist; //拿到后台返回的retlist
        total.value = resp.data.total;
      });
    }
    onMounted(()=>{
      sync_data()
    })
    provide('columns',columns)
    provide('tableData',tableData)
    provide('total',total)
    provide('callback',sync_data)
  },
};
</script>
```

移除MainLayout.vue的Props，因为不需要了

```
<template>
  <el-row justify="center">
    <el-col :span="24">
      <breadcrumb></breadcrumb>
    </el-col>
  </el-row>
  <el-row>
    <el-col >
      <tables></tables>
    </el-col>
    <el-col>
      <paginator></paginator>
    </el-col>
  </el-row>
</template>

<script>
import Breadcrumb from './Breadcrumb.vue'
import Tables from './Tables.vue'
import Paginator from './paginator.vue'

export default {
  components:{
    Breadcrumb,
    Tables,
    Paginator,
  },

}
</script>

<style scoped>
.el-row {
  margin-bottom: 20px;
  &:last-child {
    margin-bottom: 0;
  }
```

```
}
.el-col {
  border-radius: 4px;
}

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

更新Tables组件和paginator组件的输入接收方式，改为Injiect

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';


export default {
  setup(){
    const columns = inject('columns')
    const tableData = inject('tableData')

    function field_value(obj,fields){
      //操作零时对象
      let temp_obj = obj
      //根据点来分割多级字段
```

```
      for(const item of fields.split('.')){
        //通过反射来重复获取字段
        temp_obj = Reflect.get(temp_obj,item)
      }
      return temp_obj
    }
    function handleEdit(index, row) {
      console.log(index, row);
    }
    function handleDelete(index, row) {
      console.log(index, row);
    }
    return{
      handleEdit,
      handleDelete,
      field_value,
      columns,
      tableData
    }

  }
};
</script>

<style>
</style>
```

paginator.vue

```
<template>
  <el-pagination
    background
    layout="sizes,prev, pager, next"
    :total="total"
    :page-sizes="[5, 10, 20]"
    :page-size="page_size"
    @size-change="handleSizeChange"
    v-model:currentPage="page_index"
    @current-change="handleCurrentChange"
  >
  </el-pagination>
</template>

<script>
import { ref,inject} from 'vue';
export default {
  setup(props) {
    const page_size = ref(5)
    const page_index = ref(1)

    const total = inject('total')
    const callback = inject('callback')

    function handleSizeChange(size){
      console.log('page_size',size);
      page_size.value=size
```

```
      callback(page_size.value,page_index.value)
    }

    function handleCurrentChange(index){
      console.log('current page', index);
      page_index.value=index
      callback(page_size.value,page_index.value)
    }
    return {
      page_size,
      page_index,
      handleSizeChange,
      handleCurrentChange,
      total
    }
  },
};
</script>
```