RuoYi

后台手册

分页实现

- 前端基于 `element` 封装的分页组件 pagination
- 后端基于 `mybatis` 的轻量级分页插件pageHelper

前端调用实现

1、前端定义分页流程

```js
// 一般在查询参数中定义分页变量
queryParams: {
  pageNum: 1,
  pageSize: 10
},

// 页面添加分页组件，传入分页变量
<pagination
  v-show="total>0"
  :total="total"
  :page.sync="queryParams.pageNum"
  :limit.sync="queryParams.pageSize"
  @pagination="getList"
/>

// 调用后台方法，传入参数 获取结果
listUser(this.queryParams).then(response => {
    this.userList = response.rows;
    this.total = response.total;
  }
);

```

后台逻辑实现

参考后台逻辑实现

RuoYi

导入导出

在实际开发中经常需要使用导入导出功能来加快数据的操作。在项目中可以使用注解来完成此项功能。 在需要被导入导出的实体类属性添加 `@Excel` 注解，目前支持参数如下：

注解参数说明

参数	类型	默认值	
sort	int	Integer.MAX_VALUE	导出时在e 前
name	String	空	导出到Exc
dateFormat	String	空	日期格式,
dictType	String	空	如果是字典 type值 (如
readConverterExp	String	空	读取内容转 女,2=未知
separator	String	,	分隔符，读
scale	int	-1	BigDecima 启BigDeci
roundingMode	int	BigDecimal.ROUND_HALF_EVEN	BigDecima 认:BigDec
celltype	Enum	Type.STRING	导出类型
height	String	14	导出时在e 位为字符
width	String	16	导出时在e 为字符
suffix	String	空	文字后缀,
defaultValue	String	空	当值为空时
prompt	String	空	提示信息

headerBackgroundColor	Enum	IndexedColors.GREY_50_PERCENT	导出列头背 IndexedC
headerColor	Enum	IndexedColors.WHITE	导出列头字 IndexedC
backgroundColor	Enum	IndexedColors.WHITE	导出单元格 IndexedC
color	Enum	IndexedColors.BLACK	导出单元格 IndexedC
targetAttr	String	空	另一个类中 取,以小数
isStatistics	boolean	false	是否自动统 统计数据总
type	Enum	Type.ALL	字段类型 出；2：仅
align	Enum	HorizontalAlignment.CENTER	导出对齐方 Horizonta
handler	Class	ExcelHandlerAdapter.class	自定义数据
args	String[]	{}	自定义数据

导出实现流程

1、前端调用方法（参考如下）

```js
// 查询参数 queryParams
queryParams: {
  pageNum: 1,
  pageSize: 10,
  userName: undefined
},

// 导出接口exportUser
import { exportUser } from "@/api/system/user";

/** 导出按钮操作 */
```

RuoYi

```
      confirmButtonText: "确定",
      cancelButtonText: "取消",
      type: "warning"
    }).then(function() {
      return exportUser(queryParams);
    }).then(response => {
      this.download(response.msg);
    }).catch(function() {});
}
```

2、添加导出按钮事件

```html
<el-button
  type="warning"
  icon="el-icon-download"
  size="mini"
  @click="handleExport"
>导出</el-button>
```

3、在实体变量上添加@Excel注解

```java
@Excel(name = "用户序号", prompt = "用户编号")
private Long userId;

@Excel(name = "用户名称")
private String userName;

@Excel(name = "用户性别", readConverterExp = "0=男,1=女,2=未知")
private String sex;

@Excel(name = "用户头像", cellType = ColumnType.IMAGE)
private String avatar;

@Excel(name = "帐号状态", dictType = "sys_normal_disable")
private String status;

@Excel(name = "最后登陆时间", width = 30, dateFormat = "yyyy-MM-dd HH:mm:ss")
private Date loginDate;
```

```java
@Log(title = "用户管理", businessType = BusinessType.EXPORT)
@PreAuthorize("@ss.hasPermi('system:user:export')")
@GetMapping("/export")
public AjaxResult export(SysUser user)
{
    List<SysUser> list = userService.selectUserList(user);
    ExcelUtil<SysUser> util = new ExcelUtil<SysUser>(SysUser.class);
    return util.exportExcel(list, "用户数据");
}
```

导入实现流程

1、前端调用方法（参考如下）

```js
import { getToken } from "@/utils/auth";

// 用户导入参数
upload: {
  // 是否显示弹出层（用户导入）
  open: false,
  // 弹出层标题（用户导入）
  title: "",
  // 是否禁用上传
  isUploading: false,
  // 是否更新已经存在的用户数据
  updateSupport: 0,
  // 设置上传的请求头部
  headers: { Authorization: "Bearer " + getToken() },
  // 上传的地址
  url: process.env.VUE_APP_BASE_API + "/system/user/importData"
},

// 导入模板接口importTemplate
import { importTemplate } from "@/api/system/user";

/** 导入按钮操作 */
handleImport() {
  this.upload.title = "用户导入";
  this.upload.open = true;
},
/** 下载模板操作 */
```

RuoYi

```
    this.download(response.msg);
  });
},
// 文件上传中处理
handleFileUploadProgress(event, file, fileList) {
  this.upload.isUploading = true;
},
// 文件上传成功处理
handleFileSuccess(response, file, fileList) {
  this.upload.open = false;
  this.upload.isUploading = false;
  this.$refs.upload.clearFiles();
  this.$alert(response.msg, "导入结果", { dangerouslyUseHTMLString: true });
  this.getList();
},
// 提交上传文件
submitFileForm() {
  this.$refs.upload.submit();
}
```

2、添加导入按钮事件

```html
<el-button
  type="info"
  icon="el-icon-upload2"
  size="mini"
  @click="handleImport"
>导入</el-button>
```

3、添加导入前端代码

```html
<!-- 用户导入对话框 -->
<el-dialog :title="upload.title" :visible.sync="upload.open" width="400px">
  <el-upload
    ref="upload"
    :limit="1"
    accept=".xlsx, .xls"
    :headers="upload.headers"
    :action="upload.url + '?updateSupport=' + upload.updateSupport"
    :disabled="upload.isUploading"
    :on-progress="handleFileUploadProgress"
```

RuoYi

```
    drag
  >
    <i class="el-icon-upload"></i>
    <div class="el-upload__text">
      将文件拖到此处，或
      <em>点击上传</em>
    </div>
    <div class="el-upload__tip" slot="tip">
      <el-checkbox v-model="upload.updateSupport" />是否更新已经存在的用户数据
      <el-link type="info" style="font-size:12px" @click="importTemplate">下载
    </div>
    <div class="el-upload__tip" style="color:red" slot="tip">提示：仅允许导入“x
  </el-upload>
  <div slot="footer" class="dialog-footer">
    <el-button type="primary" @click="submitFileForm">确 定</el-button>
    <el-button @click="upload.open = false">取 消</el-button>
  </div>
</el-dialog>
```

4、在实体变量上添加@Excel注解，默认为导出导入，也可以单独设置仅导入Type.IMPORT

java

```java
@Excel(name = "用户序号")
private Long id;

@Excel(name = "部门编号", type = Type.IMPORT)
private Long deptId;

@Excel(name = "用户名称")
private String userName;

/** 导出部门多个对象 */
@Excels({
    @Excel(name = "部门名称", targetAttr = "deptName", type = Type.EXPORT),
    @Excel(name = "部门负责人", targetAttr = "leader", type = Type.EXPORT)
})
private SysDept dept;

/** 导出部门单个对象 */
@Excel(name = "部门名称", targetAttr = "deptName", type = Type.EXPORT)
private SysDept dept;
```

RuoYi

```java
@Log(title = "用户管理", businessType = BusinessType.IMPORT)
@PostMapping("/importData")
public AjaxResult importData(MultipartFile file, boolean updateSupport) throws
{
    ExcelUtil<SysUser> util = new ExcelUtil<SysUser>(SysUser.class);
    List<SysUser> userList = util.importExcel(file.getInputStream());
    LoginUser loginUser = tokenService.getLoginUser(ServletUtils.getRequest())
    String operName = loginUser.getUsername();
    String message = userService.importUser(userList, updateSupport, operName)
    return AjaxResult.success(message);
}

@GetMapping("/importTemplate")
public AjaxResult importTemplate()
{
    ExcelUtil<SysUser> util = new ExcelUtil<SysUser>(SysUser.class);
    return util.importTemplateExcel("用户数据");
}
```

提示

也可以直接到main运行此方法测试。

```java
InputStream is = new FileInputStream(new File("D:\\test.xlsx"));
ExcelUtil<Entity> util = new ExcelUtil<Entity>(Entity.class);
List<Entity> userList = util.importExcel(is);
```

自定义标题信息

参考自定义标题信息

自定义数据处理器

参考自定义数据处理器

自定义隐藏属性列

导出对象的子列表

参考导出对象的子列表

上传下载

首先创建一张上传文件的表，例如：

```
drop table if exists sys_file_info;
create table sys_file_info (
  file_id           int(11)          not null auto_increment       comment '文
  file_name         varchar(50)      default ''                    comment '文
  file_path         varchar(255)     default ''                    comment '文
  primary key (file_id)
) engine=innodb auto_increment=1 default charset=utf8 comment = '文件信息表';
```

上传实现流程

1、 el-input 修改成 el-upload

```vue
<el-upload
  ref="upload"
  :limit="1"
  accept=".jpg, .png"
  :action="upload.url"
  :headers="upload.headers"
  :file-list="upload.fileList"
  :on-progress="handleFileUploadProgress"
  :on-success="handleFileSuccess"
  :auto-upload="false">
  <el-button slot="trigger" size="small" type="primary">选取文件</el-button>
  <el-button style="margin-left: 10px;" size="small" type="success" :loading="
  <div slot="tip" class="el-upload__tip">只能上传jpg/png文件，且不超过500kb</di
</el-upload>
```

2、引入获取 token

3、 data 中添加属性

```js
// 上传参数
upload: {
  // 是否禁用上传
  isUploading: false,
  // 设置上传的请求头部
  headers: { Authorization: "Bearer " + getToken() },
  // 上传的地址
  url: process.env.VUE_APP_BASE_API + "/common/upload",
  // 上传的文件列表
  fileList: []
},
```

4、新增和修改操作对应处理 fileList 参数

```js
handleAdd() {
  ...
  this.upload.fileList = [];
}

handleUpdate(row) {
  ...
  this.upload.fileList = [{ name: this.form.fileName, url: this.form.filePath
}
```

5、添加对应事件

```js
// 文件提交处理
submitUpload() {
  this.$refs.upload.submit();
},
// 文件上传中处理
handleFileUploadProgress(event, file, fileList) {
  this.upload.isUploading = true;
},
// 文件上传成功处理
handleFileSuccess(response, file, fileList) {
```

```
    this.msgSuccess(response.msg);
}
```

下载实现流程

1、添加对应按钮和事件

```vue
<el-button
  size="mini"
  type="text"
  icon="el-icon-edit"
  @click="handleDownload(scope.row)"
>下载</el-button>
```

2、实现文件下载

```js
// 文件下载处理
handleDownload(row) {
  var name = row.fileName;
  var url = row.filePath;
  var suffix = url.substring(url.lastIndexOf("."), url.length);
  const a = document.createElement('a')
  a.setAttribute('download', name + suffix)
  a.setAttribute('target', '_blank')
  a.setAttribute('href', url)
  a.click()
}
```

权限注解

`Spring Security` 提供了 `Spring EL` 表达式，允许我们在定义接口访问的方法上面添加注解，来控制访问权限。

权限方法

`@PreAuthorize` 注解用于配置接口要求用户拥有某些权限才可访问，它拥有如下方法

hasPermi	String	验证用户是否具备某权限
lacksPermi	String	验证用户是否不具备某权限，与 hasPermi逻辑相反
hasAnyPermi	String	验证用户是否具有以下任意一个权限
hasRole	String	判断用户是否拥有某个角色
lacksRole	String	验证用户是否不具备某角色，与 isRole逻辑相反
hasAnyRoles	String	验证用户是否具有以下任意一个角色，多个逗号分隔

使用示例

其中 `@ss` 代表的是**PermissionService** 服务，对每个接口拦截并调用 `PermissionService` 的对应方法判断接口调用者的权限。

1. 数据权限示例。

```java
// 符合system:user:list权限要求
@PreAuthorize("@ss.hasPermi('system:user:list')")

// 不符合system:user:list权限要求
@PreAuthorize("@ss.lacksPermi('system:user:list')")

// 符合system:user:add或system:user:edit权限要求即可
@PreAuthorize("@ss.hasAnyPermi('system:user:add,system:user:edit')")
```

2. 角色权限示例。

```java
// 属于user角色
@PreAuthorize("@ss.hasRole('user')")

// 不属于user角色
@PreAuthorize("@ss.lacksRole('user')")

// 属于user或者admin之一
@PreAuthorize("@ss.hasAnyRoles('user,admin')")
```

> **权限提示**

公开接口

如果有些接口是不需要验证权限可以公开访问的，这个时候就需要我们给接口放行。

使用注解方式，只需要在 `Controller` 的类或方法上加入 `@Anonymous` 该注解即可

```java
// @PreAuthorize("@ss.xxxx('....')") 注释或删除掉原有的权限注解
@Anonymous
@GetMapping("/list")
public List<SysXxxx> list(SysXxxx xxxx)
{
    return xxxxList;
}
```

事务管理

参考事务管理实现

异常处理

参考异常处理实现

参数验证

参考参数验证

系统日志

参考系统日志实现

数据权限

参考数据权限实现

多数据源

参考多数据源实现

代码生成

参考代码生成实现

定时任务

参考定时任务实现

系统接口

参考系统接口实现

防重复提交

防重复提交实现

国际化支持

后台国际化流程

后台国际化流程

前端国际化流程

1、 package.json 中 dependencies 节点添加 vue-i18n

```json
"vue-i18n": "7.3.2",
```

RuoYi

```js
// index.js
import Vue from 'vue'
import VueI18n from 'vue-i18n'
import Cookies from 'js-cookie'
import elementEnLocale from 'element-ui/lib/locale/lang/en' // element-ui lang
import elementZhLocale from 'element-ui/lib/locale/lang/zh-CN'// element-ui la
import enLocale from './en'
import zhLocale from './zh'

Vue.use(VueI18n)

const messages = {
  en: {
    ...enLocale,
    ...elementEnLocale
  },
  zh: {
    ...zhLocale,
    ...elementZhLocale
  }
}

const i18n = new VueI18n({
  // 设置语言 选项 en | zh
  locale: Cookies.get('language') || 'en',
  // 设置文本内容
  messages
})

export default i18n
```

```js
// zh.js
export default {
  login: {
    title: '若依后台管理系统',
    logIn: '登录',
    username: '账号',
    password: '密码'
  },
  tagsView: {
```

RuoYi

```
    closeAll: '关闭所有'
  },
  settings: {
    title: '系统布局配置',
    theme: '主题色',
    tagsView: '开启 Tags-View',
    fixedHeader: '固定 Header',
    sidebarLogo: '侧边栏 Logo'
  }
}
```

```
// en.js
export default {
  login: {
    title: 'RuoYi Login Form',
    logIn: 'Log in',
    username: 'Username',
    password: 'Password'
  },
  tagsView: {
    refresh: 'Refresh',
    close: 'Close',
    closeOthers: 'Close Others',
    closeAll: 'Close All'
  },
  settings: {
    title: 'Page style setting',
    theme: 'Theme Color',
    tagsView: 'Open Tags-View',
    fixedHeader: 'Fixed Header',
    sidebarLogo: 'Sidebar Logo'
  }
}
```

3、在 `src/main.js` 中增量添加i18n

```
import i18n from './lang'

// use添加i18n
Vue.use(Element, {
```

RuoYi

```
new Vue({
  i18n,
})
```

4、在 src/store/getters.js 中添加language

```
language: state => state.app.language,
```

5、在 src/store/modules/app.js 中增量添加i18n

```
const state = {
  language: Cookies.get('language') || 'en'
}

const mutations = {
  SET_LANGUAGE: (state, language) => {
    state.language = language
    Cookies.set('language', language)
  }
}

const actions = {
  setLanguage({ commit }, language) {
    commit('SET_LANGUAGE', language)
  }
}
```

6、在 src/components/LangSelect/index.vue 中创建汉化组件

```
<template>
  <el-dropdown trigger="click" class="international" @command="handleSetLangua
    <div>
      <svg-icon class-name="international-icon" icon-class="language" />
    </div>
    <el-dropdown-menu slot="dropdown">
      <el-dropdown-item :disabled="language==='zh'" command="zh">
        中文
      </el-dropdown-item>
      <el-dropdown-item :disabled="language==='en'" command="en">
```

RuoYi

```
    </el-dropdown>
  </el-dropdown>
</template>

<script>
export default {
  computed: {
    language() {
      return this.$store.getters.language
    }
  },
  methods: {
    handleSetLanguage(lang) {
      this.$i18n.locale = lang
      this.$store.dispatch('app/setLanguage', lang)
      this.$message({
        message: '设置语言成功',
        type: 'success'
      })
    }
  }
}
</script>
```

7、登录页面汉化

```
<template>
  <div class="login">
    <el-form ref="loginForm" :model="loginForm" :rules="loginRules" class="log
      <h3 class="title">{{ $t('login.title') }}</h3>
      <lang-select class="set-language" />
      <el-form-item prop="username">
        <el-input v-model="loginForm.username" type="text" auto-complete="off"
          <svg-icon slot="prefix" icon-class="user" class="el-input__icon inpu
        </el-input>
      </el-form-item>
      <el-form-item prop="password">
        <el-input
          v-model="loginForm.password"
          type="password"
          auto-complete="off"
          :placeholder="$t('login.password')"
```

RuoYi

```
          </el-input>
        </el-form-item>
        <el-form-item prop="code">
          <el-input
            v-model="loginForm.code"
            auto-complete="off"
            placeholder="验证码"
            style="width: 63%"
            @keyup.enter.native="handleLogin"
          >
            <svg-icon slot="prefix" icon-class="validCode" class="el-input__icon
          </el-input>
          <div class="login-code">
            <img :src="codeUrl" @click="getCode" />
          </div>
        </el-form-item>
        <el-checkbox v-model="loginForm.rememberMe" style="margin:0px 0px 25px 0
        <el-form-item style="width:100%;">
          <el-button
            :loading="loading"
            size="medium"
            type="primary"
            style="width:100%;"
            @click.native.prevent="handleLogin"
          >
            <span v-if="!loading">{{ $t('login.logIn') }}</span>
            <span v-else>登 录 中...</span>
          </el-button>
        </el-form-item>
      </el-form>
      <!--  底部  -->
      <div class="el-login-footer">
        <span>Copyright © 2018-2019 ruoyi.vip All Rights Reserved.</span>
      </div>
    </div>
  </template>

  <script>
  import LangSelect from '@/components/LangSelect'
  import { getCodeImg } from "@/api/login";
  import Cookies from "js-cookie";
  import { encrypt, decrypt } from '@/utils/jsencrypt'
```

RuoYi

```
components: { LangSelect },
data() {
  return {
    codeUrl: "",
    cookiePassword: "",
    loginForm: {
      username: "admin",
      password: "admin123",
      rememberMe: false,
      code: "",
      uuid: ""
    },
    loginRules: {
      username: [
        { required: true, trigger: "blur", message: "用户名不能为空" }
      ],
      password: [
        { required: true, trigger: "blur", message: "密码不能为空" }
      ],
      code: [{ required: true, trigger: "change", message: "验证码不能为空" }]
    },
    loading: false,
    redirect: undefined
  };
},
watch: {
  $route: {
    handler: function(route) {
      this.redirect = route.query && route.query.redirect;
    },
    immediate: true
  }
},
created() {
  this.getCode();
  this.getCookie();
},
methods: {
  getCode() {
    getCodeImg().then(res => {
      this.codeUrl = "" + res.img;
      this.loginForm.uuid = res.uuid;
    });
  },
```

RuoYi

```
      const rememberMe = Cookies.get('rememberMe')
      this.loginForm = {
        username: username === undefined ? this.loginForm.username : username,
        password: password === undefined ? this.loginForm.password : decrypt(p
        rememberMe: rememberMe === undefined ? false : Boolean(rememberMe)
      };
    },
    handleLogin() {
      this.$refs.loginForm.validate(valid => {
        if (valid) {
          this.loading = true;
          if (this.loginForm.rememberMe) {
            Cookies.set("username", this.loginForm.username, { expires: 30 });
            Cookies.set("password", encrypt(this.loginForm.password), { expire
            Cookies.set('rememberMe', this.loginForm.rememberMe, { expires: 30
          } else {
            Cookies.remove("username");
            Cookies.remove("password");
            Cookies.remove('rememberMe');
          }
          this.$store
            .dispatch("Login", this.loginForm)
            .then(() => {
              this.loading = false;
              this.$router.push({ path: this.redirect || "/" });
            })
            .catch(() => {
              this.loading = false;
              this.getCode();
            });
        }
      });
    }
  }
};
</script>

<style rel="stylesheet/scss" lang="scss">
.login {
  display: flex;
  justify-content: center;
  align-items: center;
  height: 100%;
```

RuoYi

```
}
.title {
  margin: 0px auto 30px auto;
  text-align: center;
  color: #707070;
}

.login-form {
  border-radius: 6px;
  background: #ffffff;
  width: 400px;
  padding: 25px 25px 5px 25px;
  .el-input {
    height: 38px;
    input {
      height: 38px;
    }
  }
  .input-icon {
    height: 39px;
    width: 14px;
    margin-left: 2px;
  }
}
.login-tip {
  font-size: 13px;
  text-align: center;
  color: #bfbfbf;
}
.login-code {
  width: 33%;
  height: 38px;
  float: right;
  img {
    cursor: pointer;
    vertical-align: middle;
  }
}
.el-login-footer {
  height: 40px;
  line-height: 40px;
  position: fixed;
  bottom: 0;
  width: 100%;
```

```
  font-family: Arial;
  font-size: 12px;
  letter-spacing: 1px;
}
</style>
```

```
普通文本使用方式:  {{ $t('login.title') }}
标签内使用方式:    :placeholder="$t('login.password')"
js内使用方式       this.$t('login.user.password.not.match')
```

新建子模块

参考新建子模块

← 项目介绍 前端手册 →