

1 若依（前后端分离版）

1.1 什么是若依？

开源项目，学习开源项目的目标：

- 1、用，减少自己的工作量
- 2、学习优秀开源项目底层的编程思想，设计思路，提高自己的编程能力

官网：<http://www.ruoyi.vip/>

基础：Spring Boot + Vue

环境要求：

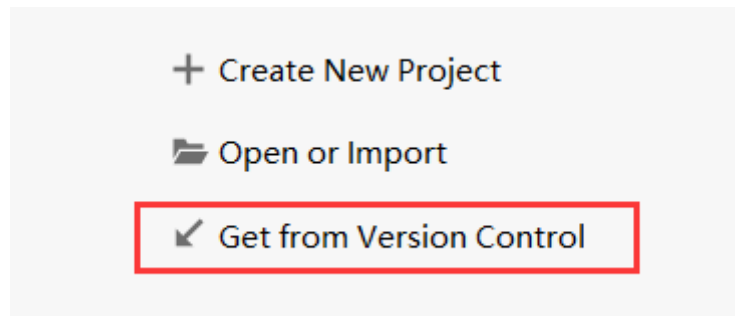
- 1、JDK 1.8+
- 2、MySQL
- 3、Redis
- 4、Maven
- 5、Vue

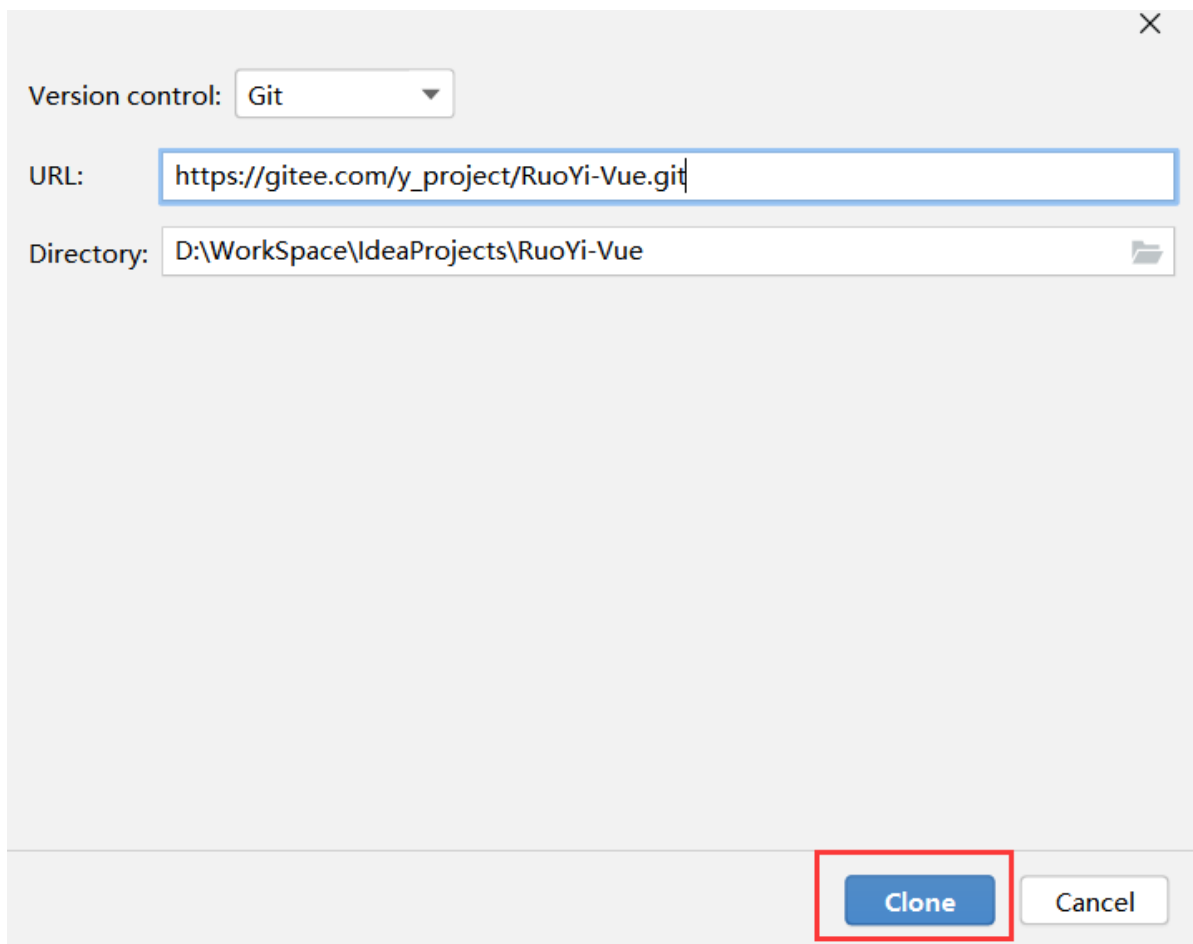
1.2 使用若依

使用开源项目的步骤：

- 1、下载并运行
- 2、看懂业务流程
- 3、进行二次开发

- 下载：





- 配置数据库 MySQL、Reids

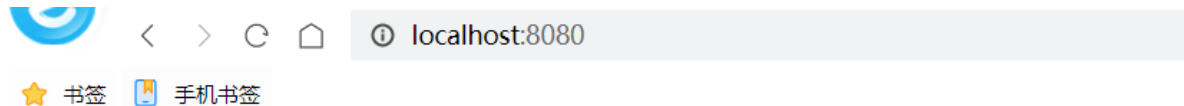
本地执行 SQL 脚本

在工程中配置数据库

```
master ▾ Ruoyi-Vue / ruoyi-admin / src / main / resources / application-druid.yml 1 2 3 4 5 6 7 8 9 10 11 # 数据源配置 spring: datasource: type: com.alibaba.druid.pool.DruidDataSource driverClassName: com.mysql.cj.jdbc.Driver druid: # 主库数据源 master: url: jdbc:mysql://localhost:3306/ry-vue?useUnicode=true&characterEncoding=utf8&zeroDateBehavior=convertToNull&useSSL=true&server username: root password: password
```

Redis 配置

```
59 # redis 配置
60 redis:
61 # 地址
62 host: localhost
63 # 端口，默认为6379
64 port: 6379
65 # 数据库索引
66 database: 0
67 # 密码
68 password:
```



欢迎使用RuoYi后台管理框架，当前版本：v3.7.0，请通过前端地址访问。

启动前端：

- 1、安装依赖 npm install
- 2、启动 npm run dev

1.3 登录

1.3.1 生成验证码

基本思路

后端生成一个表达式， $1+1=2$

$1+1=?@2$

$1+1=?$ 转成图片，传到前端进行展示

2 存入 Redis

<http://localhost/dev-api/captchalmage> 前端

Vue 获取图片，前端还是后端？

反向代理，url 请求前端，进行代理，映射到后端，解决
跨越问题

```
proxy: {  
  // detail: https://cli.vuejs.org/config/#devserver-proxy  
  [process.env.VUE_APP_BASE_API]: {  
    target: `http://localhost:8080`,  
    changeOrigin: true,  
    pathRewrite: {  
      ['^' + process.env.VUE_APP_BASE_API]: ''  
    }  
  }  
}
```

/dev-api 替换成 '' 再映射到
http://localhost:8080

<http://localhost:8080/captchalmage>

后端代码

the front end will store received-token in
Cookies,
and append the token in "Authorization"
segment in "request.js", the Wrapper
class of axios

1.3.2 登录的具体流程

后端：

- 1、校验验证码
- 2、校验用户名和密码
- 3、生成 Token

4.write into redis: login_tokens

the back end will get JWT token send
from the front end when a request
comes, use it to get uuid, and use uuid to
get LoginUser Information from redis
(which is restored in login process), it
contains username and password, use
spring security to check them, and let the
request go through if they are right.
All of this happen in "doFilterInternal".
it's a filter implemented by spring security.

使用异步任务管理器，结合线程池，实现了异步的操作
日志记录，和业务逻辑实现异步解耦合。

how to handler the authentication failures?
".exceptionHandling().authenticationEntryPoint
(unauthorizedHandler).and()" in File "SecurityConfig".

1.3.3 getInfo

获取当前用户的角色和权限信息，存储到 Vuex 中

```
*:*:*
```

1.3.4 getRouters

根据当前用户的权限获取动态路由

1.4 用户管理

流程：加载 Vue 页面 -》请求后台数据

1、getList

```
public PageData<SysUser> getList() {  
    {  
        startPage();  
        List<SysUser> list = userService.selectUserList(user);  
        return getDataTable(list);  
    }  
}
```

1、startPage()

PageHelper 中的 reasonable 对参数进行逻辑处理，保证参数的正确性，

pageNum = 0/-1, pageNum = 1

2、userService.selectUserList(user);

注解 @DataScope(deptAlias = "d", userAlias = "u")

给表设置别名的，sys_dept d, sys_user u

2、treeselect

(1) 查出所有的部门数据

(2) 组装成树状结构

buildDeptTreeSelect: 将 10 条记录组装成一个树状图

recursionFn(depts, dept);

1、先找到顶级节点，找到它的子节点

2、遍历顶级节点的子节点，再找到它的子节点

```
▶ f deptId = {Long@14365} 100
▶ f parentId = {Long@14366} 0
▶ f ancestors = "0"
▶ f deptName = "若依科技"
▶ f orderNum = "0"
▶ f leader = "若依"
▶ f phone = "15888888888"
▶ f email = "ry@qq.com"
▶ f status = "0"
▶ f delFlag = "0"
▶ f parentName = null
▶ f children = {ArrayList@14375} size = 2
▶ f searchValue = null
▶ f createBy = "admin"
```

```
▼ p data = {ArrayList@14468} size = 1
  ▼ 0 = {TreeSelect@14470}
    ▶ f id = {Long@14365} 100
    ▶ f label = "若依科技"
    ▶ f children = {ArrayList@14471} size = 2
```

点击树状图中的数据

```
// 节点单击事件
handleNodeClick(data) {
    this.queryParams.deptId = data.id;
    this.getList();
},
```

1.5 添加数据


```

handleAdd() {
    this.reset();
    this.getTreeselect();
    getUser().then(response => {
        this.postOptions = response.posts;
        this.roleOptions = response.roles;
        this.open = true;
        this.title = "添加用户";
        this.form.password = this.initPassword;
    });
},

```

reset 是表单重置

getTreeselect 获取部门树状图

getUser 获取角色和部门信息

后端 User 业务

```

@Transactional
public int insertUser(SysUser user) {
    // 新增用户信息
    int rows = userMapper.insertUser(user);
    // 新增用户岗位关联
    insertUserPost(user);
    // 新增用户与角色管理
    insertUserRole(user);
    return rows;
}

```

1.6 修改数据

```
const userId = row.userId || this.ids;
getUser(userId).then(response => {
    this.form = response.data;
    this.postOptions = response.posts;
    this.roleOptions = response.roles;
    this.form.postIds = response.postIds;
    this.form.roleIds = response.roleIds;
    this.open = true;
    this.title = "修改用户";
    this.form.password = "";
}):
```

这里的 getUser 方法除了要获取所有的岗位和角色信息之外，还要获取当前用户已经拥有的岗位和角色。

```
@PutMapping
public AjaxResult edit(@Validated @RequestBody SysUser user) {
    userService.checkUserAllowed(user);
    if (StringUtils.isEmpty(user.getPhonenumber())
        && UserConstants.NOT_UNIQUE.equals(userService.checkPhoneUnique(user)))
    {
        return AjaxResult.error("修改用户'" + user.getUserName() + "'失败，手机号码已存在");
    }
    else if (StringUtils.isEmpty(user.getEmail())
        && UserConstants.NOT_UNIQUE.equals(userService.checkEmailUnique(user)))
    {
        return AjaxResult.error("修改用户'" + user.getUserName() + "'失败，邮箱账号已存在");
    }
    user.setUpdateBy(getUsername());
    return toAjax(userService.updateUser(user));
}
```

数据库：

- 1、修改 user
- 2、重新维护 user_post 和 user_role

1.7 删除数据

```

@PreAuthorize("@ss.hasPermi('system:user:remove')")
@Log(title = "用户管理", businessType = BusinessType.DELETE)
@DeleteMapping("/{userIds}")
public AjaxResult remove(@PathVariable Long[] userIds)
{
    if (ArrayUtils.contains(userIds, getUserId()))
    {
        return error("当前用户不能删除");
    }
    return toAjax(userService.deleteUserByIds(userIds));
}

```

1.8 异步任务管理器

AsyncManager.me().execute(AsyncFactory.recordLog
infor(username, Constants.LOGIN_FAIL,
MessageUtils.message("user.password.not.match")));

通过异步任务管理器记录登录日志

- 1、AsyncManager.me() 获取一个 AsyncManager 对象
- 2、执行 execute 方法，执行任务，传入的是一个 Task 对象，实现了 Runnable 接口，是一个任务，由线程 Thread 去执行

```

public static TimerTask
recordLogininfor(final String username,
final String status, final String message,
final Object... args)
{

```

```
        final UserAgent userAgent =
UserAgent.parseUserAgentString(ServletUtils
.getRequest().getHeader("User-Agent"));
        final String ip =
IpUtils.getIpAddr(ServletUtils.getRequest()
);
        return new TimerTask()
        {
            @Override
            public void run()
            {
                String address =
AddressUtils.getRealAddressByIP(ip);
                StringBuilder s = new
StringBuilder();

                s.append(LogUtils.getBlock(ip));
                s.append(address);

                s.append(LogUtils.getBlock(username));

                s.append(LogUtils.getBlock(status));

                s.append(LogUtils.getBlock(message));
                // 打印信息到日志

                sys_user_logger.info(s.toString(), args);
                // 获取客户端操作系统
                String os =
userAgent.getOperatingSystem().getName();
```

```
// 获取客户端浏览器
String browser =
userAgent.getBrowser().getName();
// 封装对象
SysLogininfor logininfor = new
SysLogininfor();

logininfor.setUserName(username);
logininfor.setIpaddr(ip);

logininfor.setLoginLocation(address);
logininfor.setBrowser(browser);
logininfor.setOs(os);
logininfor.setMsg(message);
// 日志状态
if
(StringUtils.equalsAny(status,
Constants.LOGIN_SUCCESS, Constants.LOGOUT,
Constants.REGISTER))
{

logininfor.setStatus(Constants.SUCCESS);
}
else if
(Constants.LOGIN_FAIL.equals(status))
{

logininfor.setStatus(Constants.FAIL);
}
// 插入数据
```

```
        SpringUtils.getBean(ISysLogininforService.  
class).insertLogininfor(logininfor);  
    }  
};  
}
```

封装了登录用户的信息，执行添加操作，这里不会执行，而是将任务交给线程对象来执行。

异步任务管理器，内部定义了一个线程池，然后根据业务创建添加日志的任务，交给线程池来处理，这样做到日志和业务的抽象，解耦合，日志全部统一处理。

1.9 代码自动生成

1、创建数据表

```
use ruoyi_vue;  
create table test_user(  
    id int primary key auto_increment,  
    name varchar(11),  
    password varchar(11)  
);
```

2、系统工具 -》代码生成

● 代码生成

表名称

↓ 生成

☐ 序号

导入表

表名称

表描述

<input type="checkbox"/>	表名称	表描述
<input checked="" type="checkbox"/>	test_user	
<input type="checkbox"/>	sys_job_log	定时任务调度日志表
<input type="checkbox"/>	sys_notice	通知公告表
<input type="checkbox"/>	sys_job	定时任务调度表
<input type="checkbox"/>	sys_role_menu	角色和菜单关联表

共 18 条

3、编辑

基本信息

字段信息

生成信息

* 表名称

* 表描述

* 实体类名称

* 作者

备注

提交

返回

序号	字段列名	字段描述	物理类型
1	id	编号	int
2	name	用户名	varchar(11)
3	password	密码	varchar(11)

基本信息

字段信息

生成信息

* 生成模板

单表 (增删改查)

* 生成包路径

com.ruoyi.test

* 生成业务名

testuser

* 生成模块名

test

* 生成功能名

测试

上级菜单

系统管理

生成代码方式

☒ zip压缩包 ☐ 自定义路径

提交

返回

4、点击生成代码

预览 编辑 删除 同步 生成代码

5、解压

main (Java 后端代码)

vue (Vue 前端代码)

SQL (菜单 SQL)

6、导入代码，重启项目

如果后端抛出 404 异常，rebuild project，重新启动即可

