阅山科技 主架构说明

SpringBoot2.0，MyBatis为数据访问层，使用ApacheShiro/Jwt进行权限校验/授权， Ehcah/Redis对常用数据进行缓存。

## [技术选型](http://doc.ruoyi.vip/" \l "/standard/ry01?id=%e6%8a%80%e6%9c%af%e9%80%89%e5%9e%8b)

**1、系统环境**

* Java EE 8
* Servlet 3.0
* Apache Maven 3

**2、主框架**

* Spring Boot 2.0
* Spring Framework 5.0
* Apache Shiro 1.4
* Jwt 3.4.0

**3、持久层**

* Apache MyBatis 3.4
* Hibernate Validation 6.0
* Alibaba Druid 1.1

**4、视图层**

* Bootstrap 3.3
* Hplus 4.1
* Thymeleaf 3.0

说明：根据现阶段开发任务所选用的技术，根据不同的项目可能会略有调整，根据时间的推移，项目的推进，也会进行一定迭代。

## [主要特性](http://doc.ruoyi.vip/" \l "/standard/ry01?id=%e4%b8%bb%e8%a6%81%e7%89%b9%e6%80%a7)

* 完全响应式布局（支持电脑、平板、手机等所有主流设备）
* 一键生成功能（包括控制器、模型、视图、菜单等）
* 支持多数据源，简单配置即可实现切换。
* 支持按钮及数据权限，可自定义部门数据权限。
* 对常用js插件进行二次封装，使js代码变得简洁，更加易维护
* 完善的XSS防范及脚本过滤，杜绝XSS攻击
* Maven多项目依赖，模块及插件分项目，尽量松耦合，方便模块升级、增减模块。
* 国际化支持，服务端及客户端支持
* 完善的日志记录体系简单注解即可实现
* 缓存EhCache统一管理，支持快速切换为Redis缓存，集群Session缓存共享

后台项目结构

com.royal

├── common // 工具类

│ └── annotation // 自定义注解

│ └── config // 全局配置

│ └── constant // 通用常量

│ └── enums // 通用枚举

│ └── exception // 通用异常

│ └── json // JSON数据处理

│ └── support // 字符集处理

│ └── utils // 通用类处理

│ └── xss // XSS过滤处理

├── framework // 框架核心

│ └── aspectj // 注解实现

│ └── config // 系统配置

│ └── datasource // 数据权限

│ └── manager // 异步处理

│ └── shiro // 权限控制

│ └── util // 通用工具

│ └── web // 前端控制

├── royal-generator // 代码生成

├── royal-quartz // 定时任务

├── royal-system // 系统代码

├── royal-admin // 后台服务（前后端不分离，支持自动生成模板代码）

├── royal-xxxxxx // 其他模块

├── royal-app //前后端分离的项目

## ****[如何分页](http://doc.ruoyi.vip/" \l "/standard/ry03?id=%e5%a6%82%e4%bd%95%e5%88%86%e9%a1%b5)****

前端表格Bootstrap Table，Mybatis分页插件PageHelper

启动分页startPage()（只对该语句以后的第一个查询语句得到的数据进行分页）

@ResponseBody

public TableDataInfo list(User user)

{

startPage();

List<User> list = userService.selectUserList(user);

return getDataTable(list);

}

如果改为其他数据库需更新配置application.yml helperDialect=你的数据库

## ****[日志记录](http://doc.ruoyi.vip/" \l "/standard/ry03?id=%e6%97%a5%e5%bf%97%e8%ae%b0%e5%bd%95)****

在实际开发中，对于某些关键业务，我们通常需要记录该操作的内容  
在需要被记录日志的方法上添加@Log注解  
@Log(title = "用户管理", businessType = BusinessType.INSERT)

/\*\* 模块 \*/

String title() default "";

/\*\* 功能 \*/

BusinessType businessType() default BusinessType.OTHER;

/\*\* 操作人类别 \*/

OperatorType operatorType() default OperatorType.MANAGE;

/\*\* 是否保存请求的参数 \*/

boolean isSaveRequestData() default true;

操作类别默认为MANAGE后台用户，操作功能默认为OTHER其他，默认保存请求参数

## ****[数据权限](http://doc.ruoyi.vip/" \l "/standard/ry03?id=%e6%95%b0%e6%8d%ae%e6%9d%83%e9%99%90)****

在实际开发中，需要设置用户可以查看哪些部门的数据，一般称为数据权限  
默认用户admin拥有所有权限（userId=1）  
在需要数据权限控制方法上添加@DataScope注解  
@DataScope(tableAlias = "u")

/\*\*

\* 表的别名

\*/

String tableAlias() default "";

在mybatis添加数据范围过滤 ${params.dataScope}

## ****[多数据源](http://doc.ruoyi.vip/" \l "/standard/ry03?id=%e5%a4%9a%e6%95%b0%e6%8d%ae%e6%ba%90)****

在实际开发中，经常可能遇到在一个应用中可能需要访问多个数据库的情况  
在需要切换数据源控制方法上添加@DataSource注解  
@DataSource(value = DataSourceType.MASTER)

/\*\*

\* 切换数据源名称

\*/

public DataSourceType value() default DataSourceType.MASTER;

也可以手动控制  
DynamicDataSourceContextHolder.setDateSoureType(dataSource.value().name()); // 手动切换数据源 DynamicDataSourceContextHolder.clearDateSoureType(); // 清空数据源

## ****[导出文件](http://doc.ruoyi.vip/" \l "/standard/ry03?id=%e5%af%bc%e5%87%ba%e6%96%87%e4%bb%b6)****

在实际开发中，都需要导出对应系统数据  
在实体变量上添加@Excel注解  
@Excel(name = "用户名称")

/\*\*

\* 导出到Excel中的名字.

\*/

public abstract String name();

/\*\*

\* 提示信息

\*/

public abstract String prompt() default "";

/\*\*

\* 设置只能选择不能输入的列内容.

\*/

public abstract String[] combo() default {};

/\*\*

\* 是否导出数据,应对需求:有时我们需要导出一份模板,这是标题需要但内容需要用户手工填写.

\*/

public abstract boolean isExport() default true;

/\*\*

\* 用户导出：在Controller添加export导出方法

\*/

public AjaxResult export(User user)

{

List<User> list = userService.selectUserList(user);

ExcelUtil<User> util = new ExcelUtil<User>(User.class);

return util.exportExcel(list, "user");

}

## ****[代码生成](http://doc.ruoyi.vip/" \l "/standard/ry03?id=%e4%bb%a3%e7%a0%81%e7%94%9f%e6%88%90)****

大部分项目里其实有很多代码都是重复的，几乎每个基础模块的代码都有增删改查的功能，而这些功能都是大同小异，如果这些功能都要自己去写，将会大大浪费我们的精力降低效率。所以这种重复性的代码可以使用代码生成。

1. 修改代码生成配置  
   编辑resources目录下的application.yml  
   author: 开发者姓名，生成到类注释上  
   packageName: 默认生成包路径  
   autoRemovePre: 是否自动去除表前缀  
   tablePrefix: 表前缀
2. 新建数据库表结构（含注释）
3. drop table if exists sys\_test;
4. create table sys\_test (
5. test\_id int(11) auto\_increment comment '测试id',
6. test\_name varchar(30) default '' comment '测试名称',
7. primary key (test\_id)

) engine=innodb auto\_increment=1 default charset=utf8 comment = '测试表';

1. 登录系统-系统工具 -> 代码生成  
   找到sys\_test，点击生成代码会得到一个royal.zip  
   执行sql文件 覆盖文件到对应目录即可

## 图片Base64上传

需要在对应的application.yml配置xss排除链接，否则会接收不到base64

Royal-app

Royal-app主要用于前后端分离以及提供给第三方的接口。采用jwt+shiro进行token分发，权限校验

如果有些资源登入用户和游客看到的内容是不同的

Controller中可以通过 subject.isAuthenticated() 来判断用户是否登入

如果有些资源只有登入用户才能访问，我们只需要在方法上面加上 @RequiresAuthentication 注解即可

@RequiresAuthentication

若需要对权限进行控制，则使用注解 @RequiresPermissions("test:menu:add") 括号内严格按标准编写

Royal-app都权限校验这块代码没写：实际上在admin项目中，已经有一套权限管理模块。在royal-app中之所以不和他用同一套，是考虑到，admin的数据库是后台操作的数据库，实际开发过程中，很可能app项目和admin项目不是同一套数据库甚至权限的管理模式都可能不一样，等实际开发等适合，再根据实际情况来决定是否直接把admin等权限模块copy过来使用，还是重新写一套