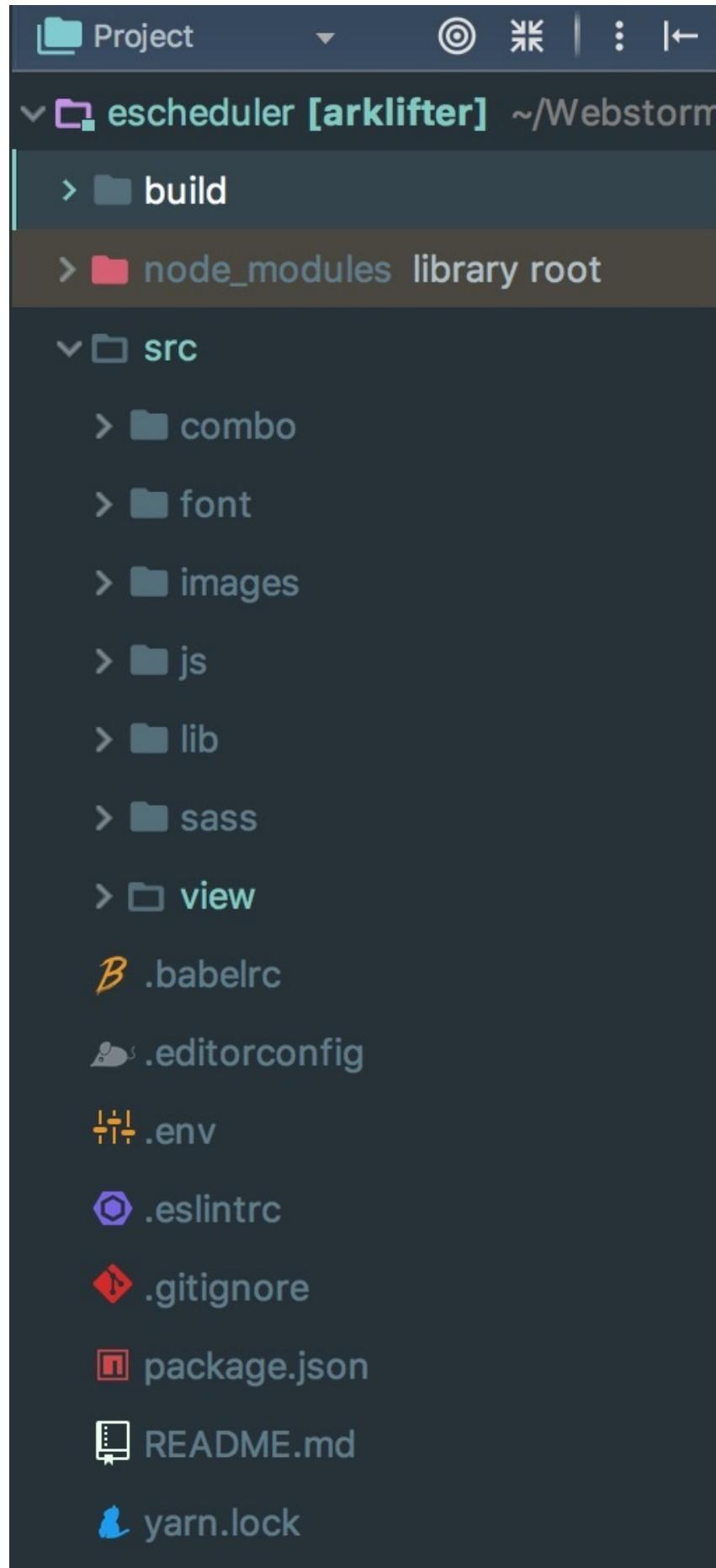


目录

Easyscheduler简介	1.1
前端开发文档	1.2
环境搭建	1.2.1
项目目录结构	1.2.2
系统功能模块	1.2.3
路由和状态管理	1.2.4
规范	1.2.5
接口列表	1.2.6
扩展开发	1.2.7
前端部署文档	1.3
前端项目环境构建及编译	1.3.1
安装及配置	1.3.2
项目生产环境Nginx配置	1.3.3
前端项目发布	1.3.4
问题	1.3.5

产品介绍

产品介绍产品介绍产品介绍产品介绍产品介绍产品介绍产品介绍产品介绍产品介绍产品介绍
产品介绍产品介绍产品介绍产品介绍产品介绍产品介绍产品介绍产品介绍产品介绍产品介绍产品介绍
产品介绍产品介绍产品介绍产品介绍产品介绍产品介绍产品介绍产品介绍产品介绍产品介绍产品介绍
产品介绍产品介绍产品介绍产品介绍产品介绍产品介绍产品介绍产品介绍产品介绍产品介绍



aaaaaa

前端部署文档

前端项目环境构建及编译

Escheduler项目前端技术栈

Vue + es6 + Ans-ui + d3 + jsplumb + lodash

开发环境

Node

- **node安装**

Node包下载 (注意版本 8.9.4) <https://nodejs.org/download/release/v8.9.4/>

- **拉取前端项目到本地**

项目git仓库地址 git@git.analysys.cn:analysys_changsha/escheduler.git

- **前端项目构建**

用命令行模式 cd 进入 escheduler 项目目录并执行 npm install 拉取项目依赖包

如果 npm install 速度非常慢

可以转淘宝镜像命令行输入 npm install -g cnpm --registry=https://registry.npm.taobao.org

运行 cnpm install

！！！ 这里特别注意 项目如果在拉取依赖包的过程中报 " node-sass error " 错误, 请在执行完后再次执行以下命令

npm install node-sass --unsafe-perm //单独安装node-sass依赖

项目根目录创建 .env 为后缀名的文件并输入

```
# 前端代理的接口地址
API_BASE = http://192.168.220.204:12345

# 如果您需要用ip访问项目可以把 "#" 号去掉
#DEV_HOST = 192.168.6.132
```

运行

- npm start 项目开发环境 (启动后访问地址 <http://localhost:8888/#/>)

- `npm run build` 项目打包 (打包后根目录会创建一个名为dist文件夹，用于发布线上Nginx)

安装及配置

(1-1) Nginx安装

安装 `wget http://nginx.org/download/nginx-1.10.1.tar.gz`

Nginx的配置及运行需要pcre、zlib等软件包的支持，因此应预先安装这些软件的开发包（devel），以便提供相应的库和头文件，确保Nginx的安装顺利完成。

```
[root@nginx ~]# service iptables stop
[root@nginx ~]# setenforce 0
[root@nginx ~]# mount /dev/cdrom /mnt/
[root@nginx ~]# vim /etc/yum.repos.d/yum.repo
[base]
name=RedHat Enterprise Linux Server
baseurl=file:///mnt/Packages
gpgcheck=0
[root@nginx ~]# yum -y install pcre-devel zlib-devel openssl-devel
```

(1-2) 创建运行用户、组

Nginx服务程序默认以nobody身份运行，建议为其创建专门的用户账号，以便更准确地控制其访问权限，增加灵活性、降低安全风险。如：创建一个名为nginx的用户，不建立宿主目录，也禁止登录到shell环境。

```
[root@nginx ~]# useradd -M -s /sbin/nologin escheduler
```

(1-3) 编译安装nginx

释放nginx源码包

```
[root@nginx ~]# tar xf nginx-1.6.2.tar.gz -C /usr/src/
```

编译前配置

```
[root@nginx ~]# cd /usr/src/nginx-1.6.2/
[root@nginx nginx-1.6.2]# ./configure --prefix=/usr/local/nginx --user=escheduler --
-group=escheduler --with-http_stub_status_module --with-http_ssl_module --with-http
_flv_module --with-http_gzip_static_module
```

注：配置前可以参考 `./configure --help`给出说明

<code>--prefix</code>	设定Nginx的安装目录
-----------------------	--------------

```
--user和--group 指定Nginx运行用户和组
--with-http_stub_status_module 启用http_stub_status_module模块以支持状态统计
--with-http_ssl_module 启用SSL模块
```

错误

```
[root@centos nginx-1.6.2]# ./configure --prefix=/usr/local/nginx --user=esched
uler --group=escheduler --with-http_stub_status_module --with-http_ssl_module
--with-http_flv_module --with-http_gzip_static_module
checking for OS
+ Linux 2.6.32-431.el6.i686 i686
checking for C compiler ... not found
./configure: error: C compiler cc is not found
```

解决方法

```
yum -y install gcc gcc-c++
```

编译 安装

```
[root@nginx nginx-1.6.2]# make && make install
```

为了使Nginx服务器的运行更加方便，可以为主程序nginx创建链接文件，以便管理员直接执行nginx命令就可以调用Nginx的主程序。

```
[root@nginx nginx-1.6.2]# ln -s /usr/local/nginx/sbin/nginx /usr/local/bin/
[root@nginx nginx-1.6.2]# ll /usr/local/bin/nginx
lrwxrwxrwx 1 root root 27 12-29 07:24 /usr/local/bin/nginx -> /usr/local/nginx/sbin
/nginx
```

Nginx的运行控制 与Apache的主程序httpd类似， Nginx的主程序也提供了"-t"选项用来对配置文件进行检查，以便找出不当或错误的配置。配置文件nginx.conf默认位于安装目录/usr/local/nginx/conf/目录中。若要检查位于其他位置的配置文件，可使用"-c"选项来指定路径。

```
root@nginx conf]# nginx -t
nginx: the configuration file /usr/local/nginx/conf/nginx.conf syntax is ok
nginx: configuration file /usr/local/nginx/conf/nginx.conf test is successful
```

启动、停止Nginx 直接运行nginx即可启动Nginx服务器，这种方式将使用默认的配置文件，若要改用其他配置文件，需添加"-c 配置文件路径"选项来指定路径。需要注意的是，若服务器中已安装有httpd等其他WEB服务软件，应采取措施（修改端口，停用或卸载）避免冲突。

```
[root@nginx conf]# chown -R escheduler:escheduler /usr/local/nginx
/usr/local/nginx/conf/nginx.conf
```

```
[root@nginx conf]# netstat -anpt |grep :80
[root@nginx conf]# nginx
[root@nginx conf]# netstat -anpt |grep :80
tcp        0      0 0.0.0.0:80          0.0.0.0:*                  LISTEN
6810/nginx: master
```

通过检查 Nginx程序的监听状态，或者在浏览器中访问此WEB服务（默认页面将显示"Welcome to nginx!"），可以确认Nginx服务是否正常运行。

```
[root@nginx ~]# yum -y install elinks
[root@nginx ~]# elinks --dump http://localhost
Welcome to nginx!
```

主程序Nginx支持标准的进程信号，通过kill或者killall命令传送

HUP	重载配置	等同于-1
QUIT	退出进程	等同于-3
KILL	杀死进程	

```
[root@nginx ~]# killall -s HUP nginx
[root@nginx ~]# killall -s QUIT nginx
[root@nginx ~]# netstat -anpt |grep :80
```

当Nginx进程运行时，PID号默认存放在logs/目录下的nginx.pid文件中，因此若改用kill命令，也可以根据nginx.pid文件中的PID号来进行控制。为了使Nginx服务的启动、停止、重载等操作更加方便，可以编写Nginx服务脚本，并使用chkconfig和service工具来进行管理，也更加符合RHEL系统的管理习惯。

```
[root@nginx ~]# vim /etc/init.d/nginx
```

脚本一

```
#!/bin/bash
# chkconfig: 2345 99 20
# description: Nginx Server Control Script
PROG="/usr/local/nginx/sbin/nginx"
PIDF="/usr/local/nginx/logs/nginx.pid"
case "$1" in
start)
    $PROG
;;
stop)
    kill -s QUIT $(cat $PIDF)
;;
restart)
    $0 stop
    $0 start
;;
esac
```

```

reload)
    kill -s HUP $(cat $PIDF)
;;
*)
    echo "Usage: $0 (start|stop|restart|reload)"
    exit 1
esac
exit 0

[root@nginx ~]# chmod +x /etc/init.d/nginx
[root@nginx ~]# chkconfig --add nginx
[root@nginx ~]# chkconfig nginx on
[root@nginx ~]# chkconfig --list nginx
nginx           0:关闭    1:关闭    2:启用    3:启用    4:启用    5:启用    6:关闭

```

报错的话： /usr/local/nginx/sbin/nginx -c /usr/local/nginx/conf/nginx.conf

这样就可以通过nginx脚本来启动、停止、重启、重载Nginx服务器了。

(2-1) root安装

安装epel源 yum install epel-release -y

安装Nginx yum install nginx -y

命令

- 启用 systemctl enable nginx
- 重启 systemctl restart nginx
- 状态 systemctl status nginx

项目生产环境配置

创建静态页面存放目录

```
mkdir /data2_4T/escheduler_front/escheduler/server
```

配置文件地址

```
/etc/nginx/conf.d/default.conf
```

配置信息

```

server {
    listen      8888; # 访问端口

```

```
server_name localhost;
#charset koi8-r;
#access_log /var/log/nginx/host.access.log main;
location / {
    root /data2_4T/escheduler_front/escheduler/server; # 静态文件目录
    index index.html index.html;
}
location /escheduler {
    proxy_pass http://192.168.220.181:12345; # 接口地址
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header x_real_ipP $remote_addr;
    proxy_set_header remote_addr $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_http_version 1.1;
    proxy_connect_timeout 4s;
    proxy_read_timeout 30s;
    proxy_send_timeout 12s;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";
}
#error_page 404 /404.html;
# redirect server error pages to the static page /50x.html
#
error_page 500 502 503 504 /50x.html;
location = /50x.html {
    root /usr/share/nginx/html;
}
}
```

重启Nginx服务

```
systemctl restart nginx
```

前端项目发布

前端在开发环境（dev）中运行 `npm run build` 命令，生成打包文件（dist）包

再拷贝到服务器 `/data2_4T/escheduler_front/escheduler/server` （服务器静态页面存放目录）

访问地址 `http://localhost:8888/#/`

问题

1. 上传文件大小限制

编辑配置文件 `vi /etc/nginx/nginx.conf`

```
# 更改上传大小  
client_max_body_size 1024m
```

前端开发文档

技术选型

Vue mvvm框架
Es6 ECMAScript 6.0
Ans-ui Analysys-ui
D3 可视化库图表库
Jspqlumb 连线插件库
Lodash 高性能的 JavaScript 实用工具库

项目目录结构

build 打包及开发环境项目的一些webpack配置
node_modules 开发环境node依赖包
src 项目所需文件
 src => combo 项目第三方资源本地化 npm run combo 具体查看 build/combo.js
 src => font 字体图标库可访问 <https://www.iconfont.cn> 进行添加 注意：字体库用自己的二次开发需要重新引入自己的库 src/sass/common/_font.scss
 src => images 公共图片存放
 src => js js/vue
 src => lib 公司内部组件（公司组件库开源后可删掉）
 src => sass sass文件 一个页面对应一个sass文件
 src => view 页面文件 一个页面对应一个html文件

> 项目采用vue单页面应用(SPA)开发
- 所有页面入口文件在 `src/js/conf/\${对应页面文件名 => home}` 的 `index.js` 入口文件
- 对应的sass文件则在 `src/sass/conf/\${对应页面文件名 => home}/index.scss`
- 对应的html文件则在 `src/view/\${对应页面文件名 => home}/index.html`

公共模块及util src/js/module

components => 内部项目公共组件

download => 下载组件

echarts => 图表组件

filter => 过滤器和vue管道

i18n => 国际化

io => io请求封装 基于axios

mixin => vue mixin 公共部分 用于disabled操作

permissions => 权限操作

util => 工具

系统功能模块

首页 => <http://localhost:8888/#/home>

项目管理 => <http://localhost:8888/#/projects/list>

- | 项目首页
- | 工作流
 - 工作流定义
 - 工作流实例
 - 任务实例

资源管理 => <http://localhost:8888/#/resource/file>

- | 文件管理
- | UDF管理
 - 资源管理
 - 函数管理

数据源管理 => <http://localhost:8888/#/datasource/list>

安全中心 => <http://localhost:8888/#/security/tenant>

- | 租户管理
- | 用户管理
- | 告警组管理
 - master
 - worker

用户中心 => <http://localhost:8888/#/user/account>

文档采用gitbook生成

中文文档 => `src/view/docs/zh_CN`

英文文档 => `src/view/docs/en_US`

路由和状态管理

项目 `src/js/conf/home` 下分为

`pages` => 路由指向页面目录

路由地址对应的页面文件

`router` => 路由管理

vue的路由器，在每个页面的入口文件`index.js` 都会注册进来 具体操作: <https://router.vuejs.org/zh/>



`store` => 状态管理

每个路由对应的页面都有一个状态管理的文件 分为:

`actions` => `mapActions` => <https://vuex.vuejs.org/zh/guide/actions.html>

`getters` => `mapGetters` => <https://vuex.vuejs.org/zh/guide/getters.html>

`index` => 入口

`mutations` => `mapMutations` => <https://vuex.vuejs.org/zh/guide/mutations.html>

`state` => `mapState` => <https://vuex.vuejs.org/zh/guide/state.html>

具体操作: <https://vuex.vuejs.org/zh/>

规范

接口列表

扩展开发