链滴

ant-design-admin 开发之旅 (1)-- 先用起来再说

作者：biristone

原文链接：https://ld246.com/article/1520749735162

来源网站：链滴

1、 安装软件
下载 node.js8.x 及以上（推荐 v8.1.2），然后安装并通过命令行检查 node --version。下载 we
storm 2017.3 版，并输入 <a href="https://ld246.com/forward?goto=http%3A%2F%2Fidea.co
ebeta.cn" target="_blank" rel="nofollow ugc">http://idea.codebeta.cn 注册。
**1、 **环境搭建
克隆项目文件:
git clone <a href="https://ld246.com/forward?goto=https%3A%2F%2Fgithub.com%2Fzui
dea%2Fantd-admin.git" target="_blank" rel="nofollow ugc">https://github.com/zuiidea/antd
admin.git
进入目录安装依赖:
#开始前请确保没有安装 roadhog、webpack 到 NPM 全局目录
npm i 或者 yarn install
编译运行：
npm run build:dll #第一次 npm run dev 时需运行此命令，使开发时编译更快
npm run dev
打开 http://localhost:8000
目录结构
├── /dist/ # 项目输出目录
├── /src/ # 项目源码目录
│ ├── /public/ # 公共文件，编译时 copy 至 dist 目录
│ ├── /components/ # UI 组件及 UI 相关方法
│ │ ├── skin.less # 全局样式
│ │ └── vars.less # 全局样式变量
│ ├── /routes/ # 路由组件
│ │ └── app.js # 路由入口
│ ├── /models/ # 数据模型
│ ├── /services/ # 数据接口
│ ├── /themes/ # 项目样式
│ ├── /mock/ # 数据 mock
│ ├── /utils/ # 工具函数
│ │ ├── config.js # 项目常规配置
│ │ ├── menu.js # 菜单及面包屑配置
│ │ ├── config.js # 项目常规配置
│ │ ├── request.js # 异步请求函数
│ │ └── theme.js # 项目需要在 js 中使用到样式变量
│ ├── route.js # 路由配置
│ ├── index.js # 入口文件
│ └── index.html
├── package.json # 项目信息
├── .eslintrc # Eslint 配置
└── .roadhogrc.js # roadhog 配置
**2、 **跨域和普通接口调用
项目自身是没有对外接口调用的，调用的数据都是 mock 数据，使用 roadhog 模拟接口,mock
口和数据都保存在 ant-design-pro-master/.roadhogrc.mock.js 中。
接口调用代码: ant-design-pro-master/src/services/api.js
比如登录接口 fakeAccountLogin，原来调用的是 mock 数据:
export async function fakeAccountLogin(params) {
return request('/api/login/account', {
method: 'POST',
body: params,
});
}
其中/api/login/account 就是.roadhogrc.mock.js 里提供的 mock 接口，如果要调用本地的接
，做如下改动:

export async function fakeAccountLogin(params) {
return request('http://localhost:8080/xxx/accountManager/login', {
method: 'POST',
body: params,
});
}
即可。
跨域问题:
由于调用本地优购接口存在跨域问题，需要请求端(ant-design-admin)和客户端(XXX)都做调整,
nt-design-admin 需要修改 ant-design-pro-master/src/utils/request.js 文件的 request 函数如下:
/p>
export default function request(url, options) {
const defaultOptions = {
credentials: 'include',
** mode: 'cors',**
};
const newOptions = { ...defaultOptions, ...options };
if (newOptions.method === 'POST' || newOptions.method === 'PUT') {
if (!(newOptions.body instanceof FormData)) {
newOptions.headers = {
Accept: 'application/json',
'Content-Type': 'application/json; charset=utf-8',
...newOptions.headers,
};
newOptions.body = JSON.stringify(newOptions.body);
} else {
// newOptions.body is FormData
newOptions.headers = {
Accept: 'application/json',
'Content-Type': 'multipart/form-data',
'Access-Control-Allow-Origin': '*',
'Access-Control-Allow-Headers': 'X-Requested-With',
'Access-Control-Allow-Methods': 'PUT,POST,GET,DELETE,OPTIONS',
...newOptions.headers,
};
}
}
服务端登录接口示例:
打开 src/com/xxx/controller/accountManager/ManagerController.java
新增代码如下:
@Resource(name="userService")
private UserService userService;
@Resource(name="roleService")
private RoleService roleService;
@Resource(name="menuService")
private MenuService menuService;
/**

访问登录页 http://localhost:8080/xxx/accountManager/login

@return

*/
** @CrossOrigin("http://localhost:8000")**
@ApiOperation(value = "访问登录页")
@ResponseBody
@RequestMapping(value="/login",method={RequestMethod.POST},consumes "application/json;charset=UTF-8")
public ResponseEntity toLogin(@ApiParam @RequestBody(required=rue) String body)throws Exception{
Map map = new HashMap();
Map params = new HashMap();
PageData pd = new PageData();
pd = this.getPageData();
String errInfo = "";
params = GsonTools.changeGsonToMap(body);
//shiro 管理的 session
Subject currentUser = SecurityUtils.getSubject();
Session session = currentUser.getSession();
if(null != params){
String sessionCode = (String)session.getAttribute(Const.SESSION_SECURITY_CODE); //获 session 中的验证码
String USERNAME = params.get("userName");
String PASSWORD = params.get("password");
pd.put("USERNAME", USERNAME);
String passwd = new SimpleHash("SHA-1", USERNAME, PASSWORD).toString(); //密码加
pd.put("PASSWORD", passwd);
pd = userService.getUserByNameAndPwd(pd);
if(pd != null){
pd.put("LAST_LOGIN",DateUtil.getTime().toString());
userService.updateLastLogin(pd);
User user = new User();
Role role = roleService.getRoleById(String.valueOf(pd.getLong("ROLE_ID")));
user.setUSER_ID(String.valueOf(pd.getLong("USER_ID")));
user.setUSERNAME(pd.getString("USERNAME"));
user.setPASSWORD(pd.getString("PASSWORD"));
user.setNAME(pd.getString("NAME"));
user.setRIGHTS(pd.getString("RIGHTS"));
user.setROLE_ID(String.valueOf(pd.getLong("ROLE_ID")));
user.setLAST_LOGIN(pd.getString("LAST_LOGIN"));
user.setIP(pd.getString("IP"));
<pre><code class="highlight-chroma">   user.setSTATUS(pd.getString("STATUS"));
</code></pre>
user.setPhone(pd.getString("PHONE"));
if(role != null){
user.setRole(role);
}
session.setAttribute(Const.SESSION_USER, user);
session.removeAttribute(Const.SESSION_SECURITY_CODE);
//shiro 加入身份验证
Subject subject = SecurityUtils.getSubject();
UsernamePasswordToken token = new UsernamePasswordToken(USERNAME, PASSWOD);

try {
subject.login(token);
} catch (AuthenticationException e) {
<pre><code class="highlight-chroma"><span class="highlight
cl">   logger.error(e.toString(), e);
</code></pre>
errInfo = "身份验证失败！";
map.put("type", "account");
map.put("currentAuthority", "changeLoginStatus");
map.put("status", "false");
return new ResponseEntity(map,HttpStatus.OK);
}
}else{
errInfo = "usererror"; //用户名或密码有误
map.put("type", "account");
map.put("currentAuthority", "changeLoginStatus");
map.put("status", "false");
return new ResponseEntity(map,HttpStatus.OK);
}
}
if(Tools.isEmpty(errInfo)){
List allmenuList = new ArrayList();
User user = (User)session.getAttribute(Const.SESSION_USER);
if (user != null) {
User userr = (User)session.getAttribute(Const.SESSION_USERROL);
if(null == userr){
user = userService.getUserAndRoleById(user.getUSER_ID());
session.setAttribute(Const.SESSION_USERROL, user);
}else{
user = userr;
}
Role role = user.getRole();
String roleRights = role!=null ? role.getRIGHTS() : "";
//避免每次拦截用户操作时查询数据库，以下将用户所属角色权限、用户权限限都存入 session<
p>
session.setAttribute(Const.SESSION_ROLE_RIGHTS, roleRights); //将角色权限存入 session<
p>
session.setAttribute(Const.SESSION_USERNAME, user.getUSERNAME()); //放入用户名
if(null == session.getAttribute(Const.SESSION_allmenuList)){
allmenuList = menuService.listAllMenu();
if(Tools.notEmpty(roleRights)){
for(Menu menu : allmenuList){
menu.setHasMenu(RightsHelper.testRights(roleRights, menu.getMENU_ID()));
if(menu.isHasMenu()){
<pre><code class="highlight-chroma"><span class="highlight
cl">   List subMenuList = menu.getSubMenu();
</code></pre>
for(Menu sub : subMenuList){
sub.setHasMenu(RightsHelper.testRights(roleRights, sub.getMENU_ID()));
<pre><code class="highlight-chroma"><span class="highlight
cl">   }
</code></pre>
}
}

}
session.setAttribute(Const.SESSION_allmenuList, allmenuList); //菜单权限放入 session 中<
p>
}else{
allmenuList = (List)session.getAttribute(Const.SESSION_allmenuList);
}
}
errInfo = "success";
map.put("type", "account");
map.put("currentAuthority", "admin");
map.put("status", "ok");
map.put("menuList", allmenuList);
//验证成功
return new ResponseEntity(map,HttpStatus.OK);
}
return null;
}
注意和原有接口注解和返回类型的不同
**3、 **菜单数据读取
Router 决定接口调用后使用哪个 model,文件在 ant-design-pro-master/src/common/router.j
,登录时会指向 model BasicLayout ,具体如下
export const getRouterData = (app) => {
const routerConfig = {
'/': {
component: dynamicWrapper(app, ['user', 'login'], () => import('../layouts/BasicLayout'
),
},
登录后 BasicLayout 会加载菜单组件，代码 ant-design-pro-master/src/layouts/BasicLayout.j

/**

根据菜单取得重定向地址.

*/
const redirectData = [];
const getRedirect = (item) => {
if (item && item.children) {
if (item.children[0] && item.children[0].path) {
redirectData.push({
from: <code>/${item.path}</code>,
to: <code>/${item.children[0].path}</code>,
});
item.children.forEach((children) => {
getRedirect(children);
});
}
}
};
getMenuData().forEach(getRedirect);
getMenuData 是通过 import { getMenuData } from '../common/menu';调用的 menu.js 里的
demo 数据，如果要使用优购的菜单数据，就需要在登录成功时先把菜单数据存储到 localStorage，
后 BasicLayout 通过 localStorage 取出来，
在登录过程中增加存储菜单函数，具体如下：
在 model :ant-design-pro-master/src/models/login.js 中先引入 menu.js 中的存储函数:** *

import { setMenuData } from '../common/menu';
effects: {
*login({ payload }, { call, put }) {
const response = yield call(fakeAccountLogin, payload);
yield put({
type: 'changeLoginStatus',
payload: response,
});
// Login successfully
if (response.status === 'ok') {
** setMenuData(response);**
reloadAuthorized();
yield put(routerRedux.push('/'));
}
},
setMenuData****如下:
export function setMenuData(menus) {
return localStorage.setItem('menus', menus);
}
获取菜单数据：
export const getMenuData = () => formatter(localStorage.getItem('menus'));
可以看到 setMenuData 会将登录接口返回的菜单数据存到 localStorage
，BasicLayout.js 再通过 getMenuData 获取菜单数据：
/**

根据菜单取得重定向地址.

*/
const redirectData = [];
const getRedirect = (item) => {
if (item && item.children) {
if (item.children[0] && item.children[0].path) {
redirectData.push({
from: <code>/${item.path}</code>,
to: <code>/${item.children[0].path}</code>,
});
item.children.forEach((children) => {
getRedirect(children);
});
}
}
};
getMenuData().forEach(getRedirect);