Author：[杭州电子科技大学](http://www.hdu.edu.cn/) 2016级管理学院 工商管理 唐涛 16011324@hdu.edu.cn

CreateTime：2019-7-25 22:00:25

UpdateTime：

Copyright: 唐涛 [home](https://www.woaihdu.top/) 2019 © 西湖大学 [propro](http://www.proteomics.pro/)

Email：[tangtao2099@outlook.com](mailto:propro@westlake.edu.cn)

# Contribution

## TANGTAO：

### Technology：

##### ant-design

##### redux

##### dva

##### react

##### umi

##### less

# DVA

npm install dva-cli -g  
  
PS F:\java\antd-2\ant-tao1-0725\0725\0725-2121> dva -v  
dva-cli version 0.10.1  
 dva version 2.6.0-beta.6

### Login

登录界面 实现 登录和注册 ，国际化

页面

**BasicLayout.jsx**

Layout style={{ minHeight: '100vh' ,minWidth:'1100px',}

这里要限制住宽度，宽度1100px 保持最小极限宽度，否则页面会显示出错。

### *添加国际化*

npm install --save react-intl

# 运行

**作者**：[唐涛](https://www.woaihdu.top)

**创建**：2019-8-3 14:40:56

**修改**：2019-8-3 14:00:38

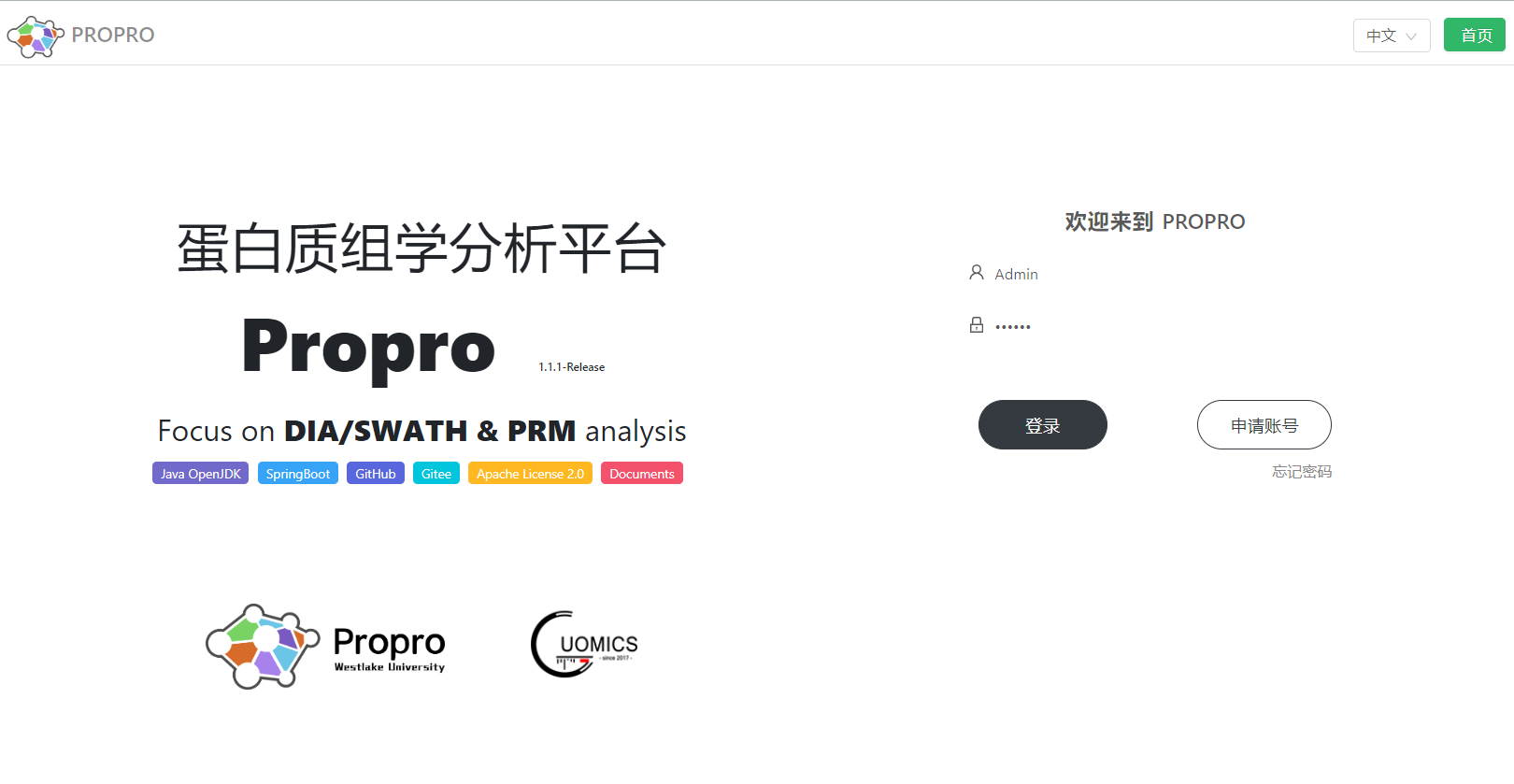
yarn install  
npm run dev

# 登录页面

**作者**：[唐涛](https://www.woaihdu.top)

**创建**：2019-8-3 14:06:48

**修改**：2019-8-3 14:00:38



# 多国语言切换模块-1(state)

**作者**：[唐涛](https://www.woaihdu.top)

**创建**：2019-7-30 23:52:34

**修改**：2019-7-30 23:52:35

### 国际化配置

/\*\*\*\*\*\*\*\*\*\*\* 国际化配置 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
/\*\*\*\*\*\*\*\*\*\*\* 国际化配置 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
// 1 引入组件  
import {IntlProvider,addLocaleData,  
 FormattedMessage, FormattedHTMLMessage} from "react-intl";  
  
// 2 引入语言包支持  
import locale\_en from 'react-intl/locale-data/en';  
import locale\_zh from 'react-intl/locale-data/zh';  
  
// 3 引入自定义的语言文件 json 格式  
import messages\_zh from "../locale/zh\_CN";  
import messages\_en from "../locale/en\_US";  
  
const messages = {  
 'zh': messages\_zh,  
 'en': messages\_en  
};  
  
// 4 设置语言支持  
addLocaleData([...locale\_en, ...locale\_zh]);  
/\*\*\*\*\*\*\*\*\*\*\* 国际化配置 end \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

### 语言结构

// src/locale/zh\_CN.js  
let language\_ch={  
 "prorpo.login":'登录',  
 "prorpo.home":'首页'  
   
}  
  
export default language\_ch;

// src/locale/en\_US.js  
let language\_en={  
 'prorpo.login':'Login',  
 'prorpo.home':'Home'  
}  
  
export default language\_en;

### 默认语言

constructor(props){  
 super(props);  
 dev\_consolelog('Initializing ...');  
  
 // navigator.language.split(/[-\_]/) zh-CN  
 // 默认从浏览器头读 但是只支持 中文 和 英语 没有读取成功 显示中文  
 let local\_language=localStorage.getItem("locale");  
 let language0= ('zh'==local\_language || 'en'==local\_language) ? local\_language : navigator.language.split(/[-\_]/)[0] ;  
 let language= ('zh'==language0 || 'en'==language0) ? language0 : 'zh' ;  
 // 把值添加到 localStorage 解决刷新问题  
 localStorage.locale = language;  
 this.state = {  
 locale: language,  
 };  
  
 dev\_consolelog('Initialization successful .');  
  
 }

### 设置默认语言

// 提取语言  
 const language=this.state.locale;  
 return (  
 <IntlProvider locale={language} messages={messages[language]} >  
 </IntlProvider>  
 );

### 设置切换语言

<Select defaultValue="中文" style={{   
 // padding:'5px 10px',  
 marginLeft:'8px',  
 height:'35px',  
 }} onChange={this.changeLanguage}>  
 <Option value="ch" >中文</Option>  
 <Option value="en">English</Option>  
 </Select>

### 切换过程

**创建时间**：2019-7-31 00:19:34

**思路**：通过 onChange() 事件触发 changeLanguage() ,从 changeLanguage() 中读取 value ，将value作为新值赋值给 this.state.locale 从而实现语言切换。

// 切换语言 触发  
 changeLanguage= e=>{  
 dev\_consolelog(`change language ${e}`);  
 this.setState({  
 locale:e,  
 });  
 // 将值添加到 localStorage  
 localStorage.locale = e;  
 }

# 多国语言切换模块-2(DVA)

**作者**：[唐涛](https://www.woaihdu.top)

**创建**：2019-8-2 03:50:43

**修改**：2019-8-2 04:07:43

### 国际化配置

/\*\*\*\*\*\*\*\*\*\*\* 国际化配置 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
/\*\*\*\*\*\*\*\*\*\*\* 国际化配置 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
// 1 引入组件  
import {IntlProvider,addLocaleData,  
 FormattedMessage, FormattedHTMLMessage} from "react-intl";  
  
// 2 引入语言包支持  
import locale\_en from 'react-intl/locale-data/en';  
import locale\_zh from 'react-intl/locale-data/zh';  
  
// 3 引入自定义的语言文件 json 格式  
import messages\_zh from "../locale/zh\_CN";  
import messages\_en from "../locale/en\_US";  
  
const messages = {  
 'zh': messages\_zh,  
 'en': messages\_en  
};  
  
// 4 设置语言支持  
addLocaleData([...locale\_en, ...locale\_zh]);  
/\*\*\*\*\*\*\*\*\*\*\* 国际化配置 end \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

### 语言结构

// src/locale/zh\_CN.js  
let language\_ch={  
 "prorpo.login":'登录',  
 "prorpo.home":'首页'  
   
}  
  
export default language\_ch;

// src/locale/en\_US.js  
let language\_en={  
 'prorpo.login':'Login',  
 'prorpo.home':'Home'  
}  
  
export default language\_en;

### 默认语言

constructor(props){  
 super(props);  
 dev\_consolelog('Initializing ...');  
  
 // navigator.language.split(/[-\_]/) zh-CN  
 // 默认从浏览器头读 但是只支持 中文 和 英语 没有读取成功 显示中文  
 let local\_language=localStorage.getItem("locale");  
 let language0= ('zh'==local\_language || 'en'==local\_language) ? local\_language : navigator.language.split(/[-\_]/)[0] ;  
 let language= ('zh'==language0 || 'en'==language0) ? language0 : 'zh' ;  
 // 把值添加到 localStorage 解决刷新问题  
 localStorage.locale = language;  
 this.state = {  
 locale: language,  
 };  
  
 dev\_consolelog('Initialization successful .');  
  
 }

###### src/models/language.js

// 语言配置 所有页面共用 这个配置留给 language 专用 用于后续扩展开发  
// navigator.language.split(/[-\_]/) zh-CN  
// 默认从浏览器头读 但是只支持 中文 和 英语 没有读取成功 显示中文  
// 先从本地数据库读  
let local\_language=localStorage.getItem("locale");  
// 第二次尝试从浏览器头取  
let language0= ( 'zh' == local\_language || 'en' == local\_language) ? local\_language : navigator.language.split(/[-\_]/)[0] ;  
// 第三次 设置默认值  
let language= ('zh'==language0 || 'en'==language0) ? language0 : 'zh' ;  
// 把值添加到 localStorage 解决刷新问题  
localStorage.locale = language;  
  
export default {  
 namespace: 'language',  
 state: {  
 // 设置 语言  
 language:localStorage.locale,  
 },  
 reducers: {  
 changeLanguage(state, { payload: new\_language }){  
 return {  
 // 更新语言配置  
 language: new\_language.language,  
 };  
 }  
 },  
 };

###### src/layout/LoginLayout.js

// state 发生改变 回调该函数 该函数返回新状态 直接导致页面刷新  
const languageStateToProps = (state) => {  
 // 先从 models 里读取  
 const language = state['language'].language;  
 return {  
 language,  
 };  
};  
  
// 语言改变触发器  
const languageDispatchToProps = (dispatch) => {  
 return {  
 changeLanguage: (language) => {  
 const action = {  
 // 触发类型  
 type: 'language/changeLanguage',  
 // 数据 payload 传入新的语言  
 payload: language,  
 };  
 // 触发  
 dispatch(action);  
 },  
 };  
};  
  
/\*\*\*\*\*\*\*\*\*\*\* 语言初始化 end \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

### 连接

// 登录  
@connect(languageStateToProps, languageDispatchToProps)  
export default class LoginLayout extends React.Component {  
...

### 获取

// 提取目标语言 从 model 中获取  
 const language=this.props.language;

### 设置默认语言

// 提取语言  
 const language=this.state.locale;  
 return (  
 <IntlProvider locale={language} messages={messages[language]} >  
 </IntlProvider>  
 );

### 设置切换语言

<Select defaultValue="中文" style={{   
 // padding:'5px 10px',  
 marginLeft:'8px',  
 height:'35px',  
 }} onChange={this.changeLanguage}>  
 <Option value="ch" >中文</Option>  
 <Option value="en">English</Option>  
 </Select>

### 切换过程

2019-8-2 04:00:52

// 切换语言 触发  
 changeLanguage= e=>{  
 dev\_consolelog(`change language ${e}`);  
 this.props.changeLanguage({  
 language: e,  
 });  
 // 将值添加到 localStorage  
 localStorage.locale = e;  
 }

*实现思路*（[TangTao](https://www.woaihdu.top)）*2019-8-2 04:32:17*

this.props.changeLanguage 会去触发 languageDispatchToProps 中定义的 changeLanguage ，它再去通过 dispatch(action); 去触发 language/changeLanguage ,再有它去设置 language: new\_language.language ，达到改变语言状态的目的。最后状态改变会回调 languageStateToProps ，最后再触发 render，达到更换界面语言的目的。

*为什么要使用 dva ？*（[TangTao](https://www.woaihdu.top)）*2019-8-2 04:32:08*

比如：为什么不通过 state 来改变语言，因为这里引入了不同的布局文件，不同布局文件之间通过 localStorage 不能很方便的设置语言，比如，一个页面选中 English ，切换另一个页面，另一个页面会重新初始化一次，把之前读取语言的流程全部重新再来一遍，整个代码要夹杂在该页面，重复性高，增加了代码量。dva 在这些页面直接起到了一个沟通桥梁的作用，脱离了子父组件之间数据传递的复杂流程，切换界面时也不用考虑记录状态，因为 render() 时不需要设置 state，而是直接从 models 读取，尤其是集成了 UMI 之后，dva 使用更加方便。models 只有在整个页面代码全部重新初始化时（刷新），才会重新执行，赋初值。

# 安装bootstrap

**作者**：[唐涛](https://www.woaihdu.top)

**创建**：2019-8-2 04:08:53

**修改**：2019-8-2 04:10:59

npm install bootstrap  
 tyarn add bootstrap

# 向服务器请求数据过程

**作者**：[唐涛](https://www.woaihdu.top)

**创建**：2019-8-3 01:02:56

**修改**：2019-8-3 02:15:01

# 登录过程

**作者**：[唐涛](https://www.woaihdu.top)

**创建**：2019-8-3 14:21:38

**修改**：2019-8-3 14:39:35

需要先熟悉 dva fetch http

## 登录初始化

*/src/pages/propro/login.js*

/\*\*\*\*\*\*\*\*\*\*\* Login View 初始化 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
/\*\*\*\*\*\*\*\*\*\*\* Login View 初始化 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
  
// state 发生改变 回调该函数 该函数返回新状态 直接导致页面刷新  
const loginStateToProps = (state) => {  
 // 先从 models 里读取 是否显示登录 当前语言  
 const login\_show = state['login'].login\_show;  
 const language = state['language'].language;  
 return {  
 login\_show,language,  
 };  
};  
  
const loginDispatchToProps = (dispatch) => {  
 return {  
 // 语言改变触发器  
 changeLogin: (login) => {  
 const action = {  
 // 触发类型  
 type: 'login/changeLogin',  
 // 数据 payload 传入新的语言  
 payload: login,  
 };  
 // 触发  
 dispatch(action);  
 },  
 // 登录触发器  
 doLogin: (login) => {  
 const action = {  
 // 触发类型  
 type: 'login/doLogin',  
 // 数据 payload 传入新的语言  
 payload: login,  
 };  
 // 触发  
 dispatch(action);  
 },  
 };  
};  
  
/\*\*\*\*\*\*\*\*\*\*\* Login View 初始化 end \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

## 点击登录

handleSubmit = e => {  
 e.preventDefault();  
 this.props.form.validateFields((err, values) => {  
 if (!err) {  
 // 执行登录  
 this.props.doLogin({  
 login: values,  
 });  
 }  
 });  
 };

## model 层接收

*path : /src/models/login.js*

let login={  
 namespace: 'login',  
 state: {  
 // 注册切换 默认显示 login  
 login\_show:true,  
 },  
 effects: {  
 \*doLogin( {payload} , sagaEffects) {  
 const { call, put } = sagaEffects;  
 const result = yield call(loginService.login,payload);  
 yield put({ type: 'doLogin\_result', payload: result });  
  
 }  
 },  
 reducers: {  
 changeLogin(state, { payload: login }){  
 return {  
 // 更新语言配置  
 login\_show: login.login,  
 };  
 },  
 doLogin\_result(state, { payload: result }){  
 console.log('doLogin\_result=====',result);  
 // return {  
 // // 更新语言配置  
 // login\_show: login.login,  
 // };  
 return {  
 login\_show:true,  
 };  
 },  
 },  
}

## 调用 server 层

export function login(data) {  
 console.log('login\_form',data);  
 let params=data.login;  
 console.log('login',params);  
 // let bodys=json2formdata(login);  
 let bodys='';  
 Object.keys(params).forEach((key) => {  
 console.log(key,params[key]);  
 bodys+=key+'='+params[key]+'&';  
 });  
   
 bodys = bodys.substr(0, bodys.length-1);  
 console.log('bodys',bodys);  
 return request('/login\_propro/test',{  
 headers: {  
 // 'content-type': 'application/json',  
 // "X-Requested-With": "XMLHttpRequest",  
 'Content-Type': 'application/x-www-form-urlencoded;charset=UTF-8'  
 },  
 method: 'POST',  
 // 发送登录数据 注意 数据未加密  
 body: bodys,  
 });  
}

## models 处理服务端返回数据

reducers: {  
 changeLogin(state, { payload: login }){  
 return {  
 // 更新语言配置  
 login\_show: login.login,  
 };  
 },  
 // 处理返回结果  
 doLogin\_result(state, { payload: result }){  
 console.log('doLogin\_result=====',result);  
 // return {  
 // // 更新语言配置  
 // login\_show: login.login,  
 // };  
 return {  
 login\_show:true,  
 };  
 },  
 },