高阶组件装饰器

装饰器函数的整个演变过程如下

```
import React from 'react';

// 1 组件原型
class Reg extends React.Component {
    render() {
        return <_Reg service={service} />;
    }
}

// 2 匿名组件
const Reg = class extends React.Component {
    render() {
        return <_Reg service={service} />;
    }
};

// 3 提参数
function inject(Comp) {
    return class extends React.Component {
        render() {
            return <Comp service={service} />;
        }
    };
}

// 继续提参数
function inject(service, Comp) {
    return class extends React.Component {
        render() {
            return <Comp service={service} />;
        }
    };
}

// 4 props
function inject(obj, Comp) {
    return class extends React.Component {
        render() {
            return <Comp {...obj} />;
        }
    };
}

// 5 柯里化
function inject(obj) {
    function wrapper(Comp) {
        return class extends React.Component {
            render() {
                return <Comp {...obj} />;
            }
        };
    }
    return wrapper;
}

// 变形
function inject(obj) {
    return function wrapper(Comp) {
        return class extends React.Component {
            render() {
                return <Comp {...obj} />;
            }
        };
    };
}

// 6 箭头函数简化
const inject = obj => {
    return Comp => {
        return class extends React.Component {
            render() {
                return <Comp {...obj} />;
            }
        };
    };
}

// 继续简化
const inject = obj => Comp => {
    return class extends React.Component {
        render() {
            return <Comp {...obj} />;
        }
    };
};
/**
 * 本质上就是被包装的组件Comp作为了返回的新匿名组件的子组件
 * 换句话说，也就是原来的组件Comp被包装成一个新的组件，增强的功能通过obj属性注入
 */

// 7 函数式组件简化
const inject = obj => Comp => {
    return props => <Comp {...obj} />;
}

const inject = obj => Comp => props => <Comp {...obj} />;

const inject = obj => Comp => props => <Comp {...obj} {...props}/>;
```

新建src/utils.js，放入以下内容

```
import React from 'react';

const inject = obj => Comp => props => <Comp {...obj} {...props}/>;

export {inject};
```

将登陆、注册组件装饰一下

```
// reg.js修改如下
import React from 'react';
import { Link, Redirect } from 'react-router-dom';
import '../css/login.css'
import { observer } from 'mobx-react';
import { userService as service } from '../service/user';
import {inject} from '../utils';

// 注意装饰器顺序
@inject({service}) //+ 装饰器注入属性
@observer
export default class Reg extends React.Component {
    validatePwd(pwd1, pwd2) {
        return pwd1.value === pwd2.value
    }

    handleClick(event) {
        event.preventDefault();
        const [name, email, password, confirm] = event.target.form;
        console.log(this.validatePwd(password, confirm)) // +要验证表单数据后才发往后台
        this.props.service.reg(name.value, email.value, password.value);
    }

    render() {
        if (this.props.service.loggedin) return <Redirect to="/" />

        let em = this.props.service.errMsg; //+ 引用这个值留作观察

        return (
            <div className="login-page">
                <div className="form">
                    <form className="register-form">
                        <input type="text" placeholder="姓名" />
                        <input type="text" placeholder="邮箱" />
                        <input type="password" placeholder="密码" />
                        <input type="password" placeholder="确认密码" />
                        <button onClick={this.handleClick.bind(this)}>注册</button>
                        <p className="message">如果已经注册 <Link to="/login">请登录</Link></p>
                    </form>
                </div>
            </div>
        );
    }

    componentDidUpdate(nextProps, nextState) { //+ 渲染后显示消息组件
        if (nextProps.service.errMsg) {
            message.info(this.props.service.errMsg, 3, () => nextProps.service.errMsg = '');
        }
    }
}
```

```
// login.js修改如下
//component/login.js
import React from 'react';
import {Link, Redirect} from 'react-router-dom';
import '../css/login.css'
import {observer} from 'mobx-react';
import {userService as service} from '../service/user';
import { message } from 'antd'; // 增加消息组件
import {inject} from '../utils';
import 'antd/lib/message/style'; // 增加样式

// 注意装饰器顺序
@inject({service}) //+ 装饰器注入属性
@observer
export default class _Login extends React.Component {
    handleClick(event) {
        event.preventDefault();
        let fm = event.target.form;
        this.props.service.login(
            fm[0].value, fm[1].value
        );
    }

    render() {
        console.log("~~~~~~~~~~~")
        if (this.props.service.loggedin) {
            return <Redirect to='/' />; // 跳转
        }

        let em = this.props.service.errMsg; // 引用这个值留作观察

        return (<div className="login-page">
            <div className="form">
                <form className="login-form">
                    <input type="text" placeholder="邮箱" />
                    <input type="password" placeholder="密码" />
                    <button onClick={this.handleClick.bind(this)}>登录</button>
                    <p className="message">还未注册? <Link to="/reg">请注册</Link></p>
                </form>
            </div>
        </div>);
    }
```

```
    componentDidUpdate(nextProps, nextState) { //+ 渲染后显示消息组件
        if (nextProps.service.errMsg) {
            message.info(this.props.service.errMsg, 3, ()=>nextProps.service.errMsg='');
        }
    }
}
```

Mobx的observer装饰器有要求，所以装饰的顺序要注意一下