## DVA 简介

dva是基于现有应用架构（redux+react-router+redux-saga等）的一层轻量封装，没有引入任何新的概念。

const app = dva({history: browserHistory,}); // 1. Initialize

app.use({}); // 2. Plugins

app.model(require('./models/example')); // 3. Model

app.router(require('./router')); // 4. Router

app.start('#root'); // 5. Start

### app.model(obj)

namespace : model 的命名空间

state: 初始state

reducers: 同步的修改状态的操作，由actions触发(state,action) => state

effects:异步的操作，并不直接修改state，由action触发，也可以调用actions。(action,{put,call,select})

subscriptions:异步的只读操作，并不直接修改state，可以调用actions。({ dispatch, history })

**put(action)和dispatch(action)**

这里effects中的put(action)等同于subscriptions中的dispatch(action)，它们的作用都是分发一个action。

yield put({ type: actionType, payload: attachedData, error: errorIfHave});

dispatch({ type: actionType, payload: attachedData, error: errorIfHave});

**call(asyncFunction)**

调用一个异步函数

const result = yield call(api.fetch, { page: 1 });

**select(function)**

从全局状态中选择数据

const count = yield select(state => state.count);

### app.router(({ history }) => routes)

import { Router, Route } from 'dva/routes';

app.router(({ history } => ({

<Router history={ history }>

<Route path="/" component={App} />

</Router>

});