


KOS

kos特点

- 简洁
- 低耦合
因为要解决多种多样的业务场景问题，要求框架本身无业务含义
- 可扩展
- 开放

基本概念

- Model
Model是View的控制器，提供View的数据源、数据操作响应（同步action，异步action）、初始化数据加载能力、表单校验和表单字段控制能力，基于kos的扩展能力redux中间件扩展所需要的配置，也将在model上承载
- View
经过kos.Wrapper包装后返回的组件，称之为view，本质上也是一个React.Component
- Wrapper
高阶函数，第一级别参数为config，第二级参数为一个React.Component包裹器，返回一个高阶组件将Model和View做了糅合
- Middleware
Redux的中间件，kos基于redux的中间件来扩展自己的能力

kos核心API

1. KOS.use(middleware)
 - 说明：新增middleware
 - 参数：redux中间件
2. KOS.start(App,Container='#main')
 - 说明：启动应用，单页应用和多页应用均调用这个API
 - 参数：
App：React.Component

Container： React.Component渲染到的DOM节点，默认是id为main的节点

3. KOS.Wrapper(config)(Component)

- 说明： 将Component使用Wrapper组件进行包装后，挂在到connect下面，用户将Model和View进行包装
- 参数：
 - config.model： 需要绑定的model，model说明详见 Model
 - config.autoLoad： 在执行到Wrapper的componentDidMount的时候，是否自动执行Model.setup方法
- 返回值： 和Model结合之后的高阶组件

Model

1. namespace

- 类型：string，
- 说明： namespace是用来区分不同Model和组件的唯一标识，不能重复，封装后的组件将使用store.getState()对象下的namespace一级的数据

2. initial

- 类型：Object
- 说明：提供初始化的默认数据

3. reducers

- 类型：Object
- 说明：提供处理同步action的逻辑，key为action.type

4. setup

- 说明：加载初始化数据时，执行的action，WrapperComponent会根据Wrapper(config)的autoLoad来确定，是否需要执行

5. reset

- 说明：reset会将model.initial下的数据，覆盖store.getState()
[namespace]的数据，默认componentDidMount的时候，会出发该
action

Util

1. Util.wrapperDispatch(dispatch,namespace)(action)

-说明：这是一个高阶函数，根据第一阶函数dispatch和namespace，执行第二阶的action的dispatch

- 参数：
dispatch：store.dispatch方法
namespace：命名空间名称
- 返回值：action

2. Util.getParam()

- 说明：获取参数，包括url后面的query和hashUrl后面的query，不包括路由的match
- 返回值：key:value形式的参数对象，例如{id:1}

3. Util.getActionType(actionType)

- 说明：拆分actionType为namespace和type
- 参数：
actionType：包含namepsace的type，例如：'page-add/save'
- 返回值：返回namespace和type组成的对象，例如上面的将返回