**MyWidget**

1、扩展接口 util, domUtil

2、resetConfig：传入object对象，修改组件中的attr

3、静态方法：

**use(query, config)**——如use(‘.searchList’, {})。可对相同class的多个元素实例化对象。返回以每个query作为element的组件实例数组。

$(query).each(function(){

list.push( new me( $.extend({element: this}, config) ) );

});

**dead(query)**——每个query生成的实例调用destroy方法销毁对象

**getWidget(query)**——返回query元素节点中的myWidget属性

**PerSearch**

**初始化**

1、调用**searchListSerilize**序列化参数

1. 获取容器：me.content = me.$('.content')
2. 设置模板：me.template 为编译过的模板，可通过配置compileTemplate（编译过的hbs文件）获取。

也可以 配置template或 获取me.$('.template').html()（未经编译的原始模板），编译后赋值给me.template

**入口**

配置autoStart，如果为true，在页面打开后自动调用searchListAjax通过ajax获取数据

**searchListSerilize**

1. 调用domUtil.serialize() 序列化.param元素中的参数。
2. 可在调用时配置param扩展自动生成的参数。
3. 参数中添加page：{begin, length}

**searchListAjax**

调用domUtil.ajax(url, data, type, callback)：格式化入参后，调用$.ajax 。

参数:

url: me.get('request')

data: domUtil.paseParam(me.get('paramName'), me.searchListParam) ，用paramName包裹searchListParam

callback(err, response)，response为返回的数据

配置mapResponse，格式化response

配置map，格式化response.data

配置size，一页显示数量

在 me.content中渲染数据

初始化分页

**searchListReload**

重新设置分页，通过**searchListSerilize**参数序列化后调用**searchListAjax**重新渲染页面

**searchListDestroyPage**销毁分页

**SearchList**

对PerSearch组件添加重载、修改、删除事件

1. 修改：'click .JS-trigger-click-editor'

调用 domUtil.http(url, data, type, callback)

url: 配置requestEditor，如通过元素属性传参，在searchList元素上 data-request-editor=""

data: 在修改按钮上添加 data-param="" 传入参数

callback(err, rtv, msg, response)

错误时：

err——( json.errors && json.errors[0] && json.errors[0].msg )||'ajax请求，系统异常！'

rtv——json.errors

成功时：

content —— json.content

err——null

rtv——content.retValue,

msg——content.message

response——content