交互式组件具有“可交互容器”和“可交互组件”两种

可交互容器提供数据

可交互组件只有数据

**可交互容器**

**组件对象配置**

import BaseComponentObject, {ComponentSettingConfig} from '../../BaseComponents/BaseComponent';

import ComponentDescribe, {componentType} from '../../ComponentDescribe'

import InteractiveContainerConfig from '../../BaseComponents/InteractiveComponent/InteractiveContainerConfig'

import IocContainer from 'Core/IocContainer';

import IComponent from './IComponent'

import IPreview from './IPreview'

import ISettingConfig from './ISettingConfig'

class ComponentObject extends BaseComponentObject{

    ComponentSettingConfigs = [

        // 指定当前组件为可交互容器

        new InteractiveContainerConfig(),

        new ComponentSettingConfig(undefined, "标签设置", IocContainer.getService(ISettingConfig))

    ];

    Component=IocContainer.getService(IComponent)

    Preview=IocContainer.getService(IPreview)

}

const componentDescribeBuilder = () => new ComponentDescribe("IECategoryLabel", new ComponentObject(), componentType.page, "IE分类标签");

export default componentDescribeBuilder;

**组件**

可交互组件的组件props具有两个属性

// 是否存在子组件

isExitChild={true}

// 子组件

ChildComponent={childs}

如下示例：

import React from 'react'

import IComponent from 'BaseCMSManage/Components/IETemplateComponents/IECategoryLabel/IComponent'

import { Link, withRouter } from 'react-router-dom'

import InteractivConfigFeature, { InteractivConfigFeatureClickItem, InteractivConfigFeatureTextItem } from 'BaseCMSManage/Components/BaseComponents/InteractiveComponent/InteractivConfigFeature'

class IECategoryLabel extends IComponent {

    interactiveComponent(datas) {

        // 配置可交互选项

        let items = [

            // text 可选项名称

            // 标签项文本 展示名称

            // 第 3 个参数是如何取得这个可选项的数据，data 简绍往后看

            new InteractivConfigFeatureTextItem('text', '标签项文本', (data) => {

                return data.displayName;

            }),

            new InteractivConfigFeatureClickItem('click', '标签项点击', (data) => {

                return () => {

                    this.props.history.push(this.createUrl(data.tagName));

                }

            }),

        ];

        return (

            <div className="d-flex flex-wrap">

                {datas.map(item => {

                    // ChildComponent 是一个组件类型，需要使用 <... />

                    // interactivConfigFeature 交互配置特征，items 即我们的配置可交互选项，item 即上面提到的 data

                    return <this.props.ChildComponent

                        interactivConfigFeature={new InteractivConfigFeature(items, item)}

                    />

                })}

            </div>

        );

    }

    render() {

        ...

        // 如果不存在子组件，则使用自己默认的实现

        if (!this.props.isExitChild) {

            return this.defaultComponent(datas);

        }

        // 否则使用子组件

        else {

            return this.interactiveComponent(datas);

        }

    }

}

IECategoryLabel.defaultProps = {

};

export default (register) => register(IComponent, withRouter(IECategoryLabel));

**可交互组件**

**组件对象配置**

import BaseComponentObject, {ComponentSettingConfig} from '../../BaseComponents/BaseComponent';

import ComponentDescribe, {componentType} from '../../ComponentDescribe'

import InteractiveComponentConfig from '../../BaseComponents/InteractiveComponent/InteractiveComponentConfig'

import IocContainer from 'Core/IocContainer';

import IComponent from './IComponent'

import ISettingConfig from './ISettingConfig'

import IPreview from './IPreview'

class ComponentObject extends BaseComponentObject {

    ComponentSettingConfigs = [

        // 添加可交互组件配置

        new InteractiveComponentConfig(),

        new ComponentSettingConfig("BtnSetting", "按钮设置", IocContainer.getService(ISettingConfig))

    ];

    Component=IocContainer.getService(IComponent);

    Preview=IocContainer.getService(IPreview);

}

const componentDescribeBuilder = () => new ComponentDescribe("IEButton", new ComponentObject(), componentType.other, "IE按钮");

export default componentDescribeBuilder;

**组件**

可交互组件的组件props具有 interactivClick （点击），interactivText （文本）两个属性

import React from 'react';

import IComponent from 'BaseCMSManage/Components/IETemplateComponents/IEButton/IComponent'

import { Button } from 'antd';

import Setting from 'BaseCMSManage/Components/IETemplateComponents/IEButton/Setting'

import {getIconType} from 'Common/AntIcons'

import {withRouter} from 'react-router-dom'

class IEButton extends IComponent {

    render() {

        let setting = new Setting(this.getSetting("BtnSetting"));

        let Icon;

        if(setting.icon){

            let IconType = getIconType(setting.icon);

            Icon = <IconType />

        }

        return (

        <Button

            icon={Icon}

            style={{backgroundColor: setting.bgcolor, color: setting.color}}

            shape={setting.shape} type={setting.btnType} size={setting.size}

            onClick={()=>{

                // 如果存在 interactivClick，则使用 interactivClick

                if(this.props.interactivClick){

                    this.props.interactivClick();

                    return;

                }

                if(setting.url.startsWith('http')){

                    window.location.href = setting.url;

                    return;

                }

                this.props.history.push(setting.url);

            }}

        >

            {/\* 如果存在 interactivText，则使用 interactivText \*/}

            {this.props.interactivText || setting.text}

        </Button>

        );

    }

}

export default (register) => register(IComponent, withRouter(IEButton));