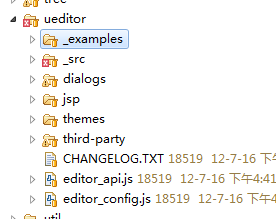
文本编辑器使用手册

本系统表单设计使用的文本编辑器是百度的开源编辑器[ueditor](http://ueditor.baidu.com)，版本是2012年5月31日发布的1.2.1.0，在此版本的基础上做了一些改进和定制开发。编辑器的所有相关文件均在src\main\webapp\js\ueditor文件夹下。

## 目录结构说明



* \_examples为一些示例页面；
* \_src为一些通用的基础js文件、插件js文件、ui的逻辑js文件；
* dialogs中为某些具有界面的功能的html、css、js文件；
* jsp是一些文件上传、图片上传和管理等页面；
* themes是css和图片资源的文件夹；
* third-party是第三方包的文件夹；
* CHANGELOG.TXT是ueditor的更新日志；
* editor\_api.js是编辑器引用哪些js的配置文件（这是在开发模式下使用该文件，在项目正式部署的时候，可以将所用到的js文件全部放到一起压缩以后进行发布）；
* editor\_config.js是编辑器的配置文件。

## 工具栏自定义按钮

要添加自定义的工具栏按钮，需要以下几个步骤：

### 添加按钮的js文件

例如我们现在需要添加针对按钮的功能，包括：新增按钮、修改按钮、删除按钮等功能，那就需要在ueditor\\_src\plugins\目录下添加input.js文件。文件的内容大致是这样的。

UE.plugins['input'] = **function**() {

**var** me = **this**;

me.commands['addinput'] = {

execCommand : **function**(cmdName) {

me.curInput=**null**;

**this**.ui.\_dialogs['addinputbutton'].open();

**return** **true**;

},

queryCommandState : **function**() {

**return** **this**.highlight ? -1 : 0;

}

};

me.commands['editinput'] = {

execCommand : **function**(cmdName) {

me.curInput=**this**.selection.getRange().getClosedNode();

**this**.ui.\_dialogs['addinputbutton'].open();

**return** **true**;

},

queryCommandState : **function**() {

**var** el = **this**.selection.getRange().getClosedNode();

**if**(!el){

**return** -1;

}

**else** **if**(el.tagName.toLowerCase()=='input'||el.tagName.toLowerCase()=='textarea'||el.tagName.toLowerCase()=='select'){

**return** **this**.highlight ? -1 : 0;

}

**return** -1;

}

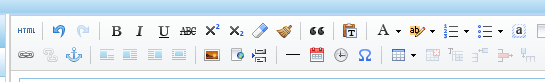
};

};

这个插件名称叫input，拥有两个功能，分别是addinput和editinput。每个功能有两个方法，都是execCommand和queryCommandState。

前者是执行这个功能，比如执行插入input的功能或者编辑input的功能（具体的功能实现逻辑需要自己来完成，不过在功能里面也可以调用其他已有的功能）。

后者是查询该功能的状态（即是否可以使用该功能）。如下图所示，有的功能呈现高亮状态，表示当前状态下可以使用；有的呈灰色，表示当前状态下不能使用该功能，而这些都是通过queryCommandState来实现的（对于下面即将要讲到的右键菜单也采用了这种机制，在不同的区域或对象上点鼠标右键会看到不同的右键菜单）。



### editor\_config.js中配置按钮

完成第一步以后，就需要在配置文件中配置好该功能了。在toolbars中添加input，这里的先后顺序会决定最终界面上的先后顺序。在labelMap中添加上对input的tooltip提示。



### editorui.js中实例化按钮

//添加按钮

editorui.input = **function** (editor,iframeUrl ,title){

iframeUrl = iframeUrl || (editor.options.iframeUrlMap||{})['input'] || iframeUrlMap['input'];

title = title || editor.options.labelMap['input'] || '';

**if**(!iframeUrl){

**return**;

}

**var** dialog = **new** editorui.Dialog({

iframeUrl: editor.ui.mapUrl(iframeUrl),

editor: editor,

className: 'edui-for-input',

title: title,

buttons: [{

className: 'edui-okbutton',

label: '确认',

onclick: **function** (){

dialog.close(**true**);

}

}, {

className: 'edui-cancelbutton',

label: '取消',

onclick: **function** (){

dialog.close(**false**);

}

}]

});

dialog.render();

editor.ui.\_dialogs['addinputbutton'] = dialog;

**var** ui = **new** editorui.Button({

editor:editor,

className : 'edui-for-input',

title : title || editor.options.labelMap['input'] || '',

onclick: **function** (){

editor.execCommand("addinput");

}

});

editor.addListener('selectionchange', **function** (t,v){

ui.setDisabled(editor.queryCommandState('addinput') == -1);

});

**return** ui;

};

这一段是实例化按钮、新增和编辑按钮界面（有些功能是有弹出的对话框作为界面的，后面会具体讲到怎么自定义对话框）的代码，这段代码应该写到ueditor\\_src\ui\editorui.js文件中。

这里主要实例化了editorui.input，其中又包括了dialog和ui的实例化，其中dialog和ui都分别指定了className: ' edui-for-input '和className : 'edui-for-input'。这些class都需要我们自己添加，添加到ueditor\themes\default\ ueditor.css中。dialog的class指定了出现的对话框的长度和宽度，ui的class指定了出现在工具栏中的按钮所使用的图标（图标使用的图片在edui-icon中已经指定，对具体功能所使用的图标只需要修正偏移量就可以了。如果觉得自带的图标太少，我们也可以自己放一些图标到资源文件夹中ueditor\themes\default\ images）。

.edui-for-input .edui-icon{

background-position: -340px -40px;

}

/\*input-dialog\*/

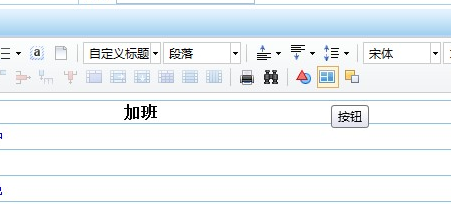
.edui-for-input .edui-dialog-content {

width:400px;

height:150px;

}

完成以上这些步骤，就基本上实现工具栏按钮的自定义了。



## 右键菜单

右键菜单，同样需要我们在plugins文件夹中添加对应的js文件。上面讲到的工具栏自定义按钮中已经添加了input.js，那我们就以自定义“编辑按钮”这个菜单为例来说明，这个右键菜单对应的是input.js中的editinput功能。

在editor\_config.js中的contextMenu后面追加右键菜单的定义。

,contextMenu : [

{

label:'修改按钮',

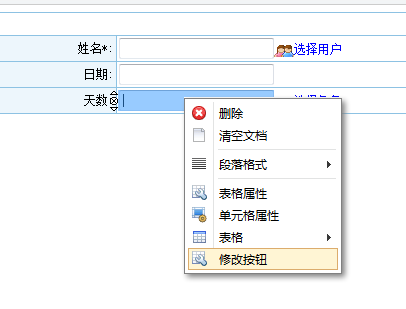
cmdName:'editinput',

icon:'editinput'

}

]

label是右键菜单上显示的按钮标签，cmdName对应了按钮的功能，icon是按钮标签前面的图标（当然前提是需要在ueditor\themes\default\ueditor.css中先定义editinput样式）。



完成这些配置以后，我们就可以在右键菜单中看到修改按钮的选项了。但是我们需要在用户选中input标签时才给出这个选项，那就需要配置editinput这个功能的queryCommandState方法了。

queryCommandState : **function**() {

**var** el = **this**.selection.getRange().getClosedNode();

**if**(!el){

**return** -1;

}

**else** **if**(el.tagName.toLowerCase()=='input'||el.tagName.toLowerCase()=='textarea'||el.tagName.toLowerCase()=='select'){

**return** **this**.highlight ? -1 : 0;

}

**return** -1;

}

这里获取到的el对象就是选中的区域，如果这个el为空，则返回-1（返回-1表示不可用，返回0表示可用），否则就判断el的标签类型，当标签类型是input、textarea、select的时候，我们继续判断当前是否在highlight状态（即是源代码状态还是在编辑状态），如果在源代码状态则返回-1，在编辑状态就返回0。

对于二级右键菜单，我们只需要在配置contextMenu的时候添加上group就可以了。

,contextMenu : [

{

label:'修改按钮',

cmdName:'editinput',

icon:'editinput'

},

'-',

{

group:'更改按钮类型',

icon:'justifyjustify',

subMenu:[

{

label:'单行文本',

cmdName:'justify',

value:'left'

},

{

label:'多行文本',

cmdName:'justify',

value:'right'

},

{

label:'按钮',

cmdName:'justify',

value:'center'

},

{

label:'单选按钮',

cmdName:'justify',

value:'justify'

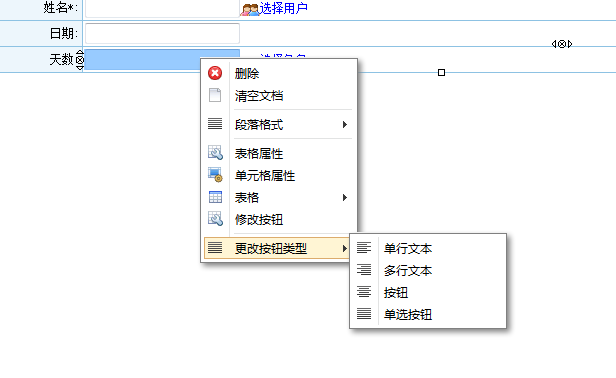
}

]

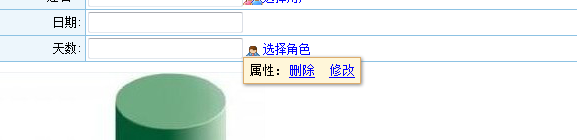
}

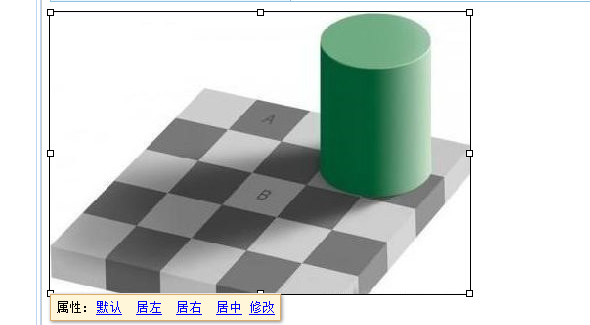
]

我们在两个右键菜单选项之间加上’-’，表示在这两个选项之间用横线分割。下面的更改按钮类型的选项是一个group，则表示这个选项有二级菜单。



## 浮动工具条





editor.addListener( 'mouseover', **function**( t, evt ) {

**var** el = evt.target || evt.srcElement;

**if**(!(el.tagName&&el.parentElement))**return**;

**var** ifEl=el.getAttribute("name")=="editable-input";

**var** ifElParent = el.parentElement.getAttribute("name") =="editable-input";

**if**(ifEl||ifElParent){

**var** html = popup.formatHtml(

'<nobr>属性: <span onclick="$$.\_onInputDelButtonClick()" class="edui-clickable">删除</span>&nbsp;&nbsp;' +

'<span onclick="$$.\_onInputEditButtonClick(event, this);" class="edui-clickable">修改</span></nobr>' );

popup.getDom( 'content' ).innerHTML = html;

popup.anchorEl = ifEl?el:el.parentElement;

popup.showAnchor( popup.anchorEl );

}

} );

editor.addListener( 'selectionchange', **function** ( t, causeByUi ) {

**if** (!causeByUi ) **return**;

**var** html = '';

**var** img = editor.selection.getRange().getClosedNode();

**if** ( img != **null** && img.tagName.toLowerCase() == 'img' ) {

**if** ( img.getAttribute( 'anchorname' ) ) {

html += popup.formatHtml(

'<nobr>属性: <span onclick=$$.\_onImgEditButtonClick(event) class="edui-clickable">修改</span>&nbsp;&nbsp;<span onclick=$$.\_onRemoveButtonClick(event) class="edui-clickable">删除</span></nobr>' );

} **else** **if** ( editor.options.imagePopup ) {

html += popup.formatHtml(

'<nobr>属性: <span onclick=$$.\_onImgSetFloat("none") class="edui-clickable">默认</span>&nbsp;&nbsp;<span onclick=$$.\_onImgSetFloat("left") class="edui-clickable">居左</span>&nbsp;&nbsp;<span onclick=$$.\_onImgSetFloat("right") class="edui-clickable">居右</span>&nbsp;&nbsp;' +

'<span onclick=$$.\_onImgSetFloat("center") class="edui-clickable">居中</span>' +

' <span onclick="$$.\_onImgEditButtonClick(event, this);" class="edui-clickable">修改</span></nobr>' );

}

}

**if** ( html ) {

popup.getDom( 'content' ).innerHTML = html;

popup.anchorEl = img || link;

popup.showAnchor( popup.anchorEl );

} **else** {

popup.hide();

}

} );

浮动窗口有两种触发方式，一种是鼠标点选时出现，另一种是鼠标移上去就出现。在ueditor\\_src\ui\ editor.js中，分别对mouseover和selectionchange事件添加监听。当符合代码中的判定条件时，就显示浮动窗口。

浮动窗口中的功能按钮，比如修改、删除等操作的代码同样在ueditor\\_src\ui\ editor.js中定义，比如针对按钮的浮动窗口有两个功能按钮，分别是删除和修改，其对应的代码为：

//处理输入框的修改

\_onInputEditButtonClick:**function**(){

**this**.hide();

editor.curInput=popup.anchorEl;

editor.ui.\_dialogs.addinputbutton.open();

},

\_onInputDelButtonClick:**function**(){

**this**.hide();

**if**(popup.anchorEl){

popup.anchorEl.parentNode.removeChild(popup.anchorEl);

}

},

## 自定义对话框

自定义的对话框，刚才1.2.3中的自定义按钮的实例化中已经实例化过dialog了。除了要实例化dialog以外，还需要以下几点：

### 创建html或者jsp页面作为对话框的界面

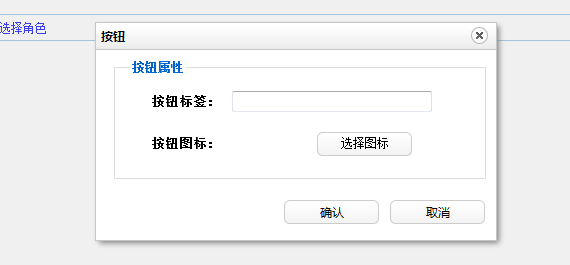
以添加按钮为例，在ueditor\dialog\目录下创建input文件夹，将界面用到的html、css、js文件放在这个目录下。

### 在editorui.js中添加对话框和html页面的对应关系

在ueditor\\_src\ui\editorui.js文件中的iframeUrlMap中添加上input和html文件的对应关系。



完成这些步骤以后，在input.js中的添加按钮或者修改按钮中打开对话框就可以看到按钮的操作界面了。



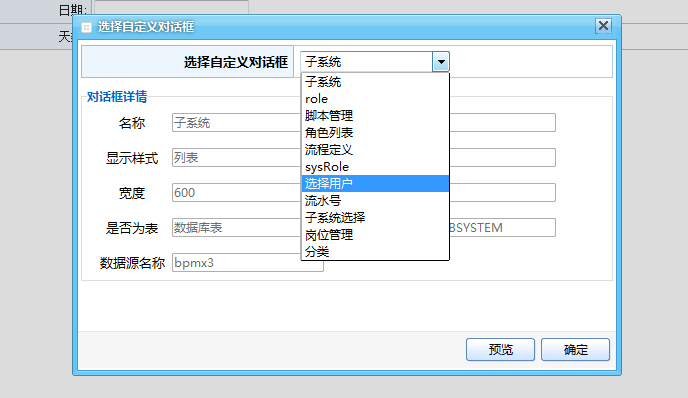
## 表单设计中编辑器的使用

以上是一些对于ueditor编辑器的定制开发的介绍，下面介绍一下在表单设计中怎么使用这个编辑器，让我们更方便的设计出功能强大的表单。

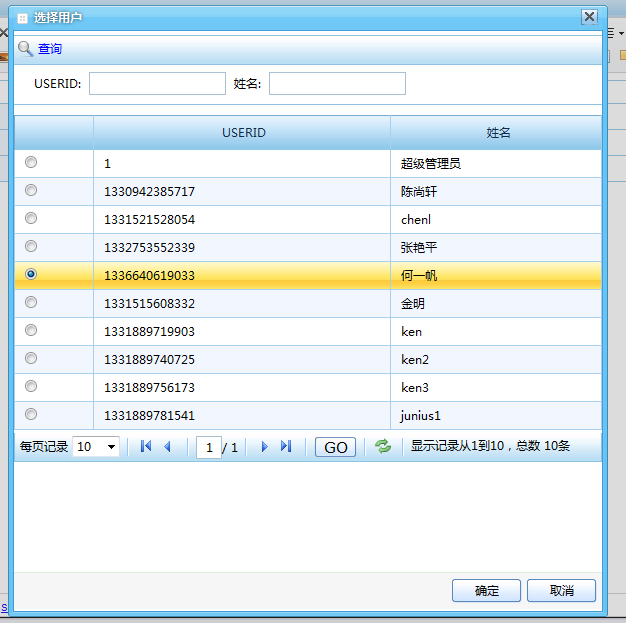
### 添加自定义对话框

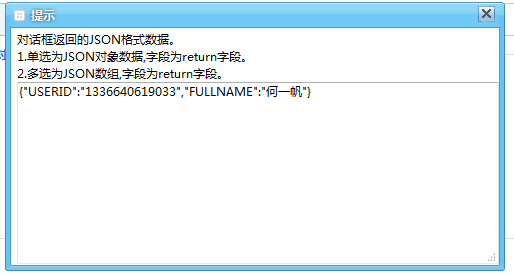
有关自定义对话框的定制可以查看《流程自定义表单文档》中的3.7节。

首先选择工具栏上的自定义对话框按钮打开选择对话框的界面：

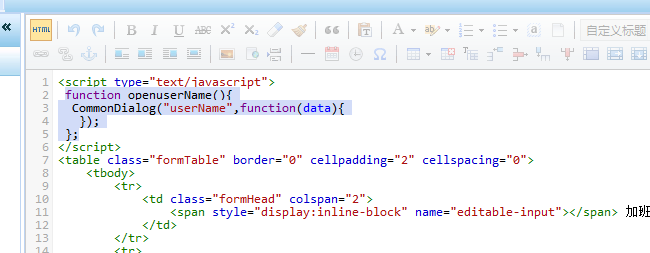


选择好对话框以后，点击预览会弹出改对话框，选择数据以后点确定，会弹出对话框来显示返回的数据。



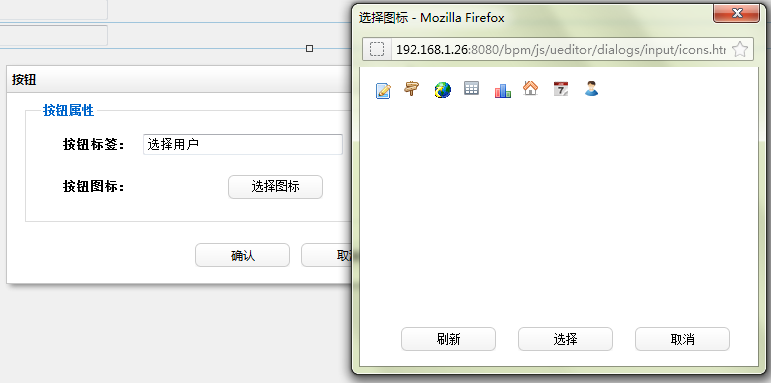


在选择对话框界面上点击确定按钮，会在表单中生成打开对话框的js方法（会自动找到表单开头的script标签，并附加到标签中已有方法的后面，没有script标签会自动创建）。



### 添加按钮并绑定事件

点击工具栏上的按钮图标打开按钮对话框，填上按钮的标签。并点击选择图标来为按钮选择一个图标，选择图标的页面中列出的图标与系统的webapp\styles\default\images\extendIcon文件夹下面的图标一一对应，如果要新增图标，只需要将图标复制到该文件夹下，并在页面上点击刷新按钮即可（如果刷新时没有显示出新增加得图标，多点一次刷新按钮，可能是浏览器缓存所致）。



选择图标以后点击确认按钮，编辑器会再光标位置插入一个带有背景图标的a标签。现在切换到源代码模式下，为该a标签绑定onclick事件。



这样点击该按钮就会弹出选择用户的自定义对话框。

### 处理回调事件

处理自定义对话框的回调事件，在用户打开选择用户的对话框，选择用户以后，再点击确认按钮就会触发回调事件，此时我们需要将回调的数据显示文本框中。就需要像下面这样写一句赋值的代码。

<script type=*"text/javascript"*>

**function** openuserName() {

CommonDialog("userName", **function**(data) {

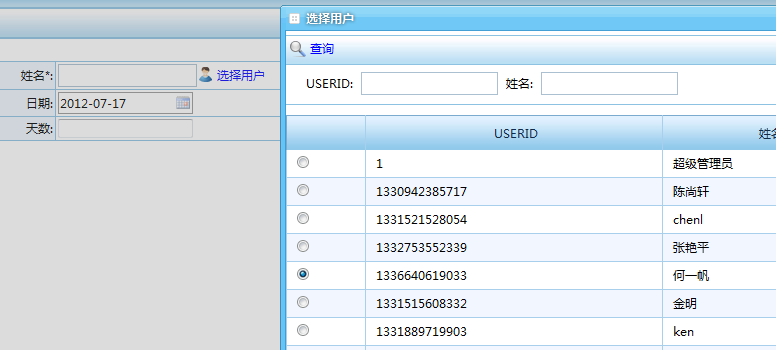
$("input[name='m:jiaban:name']").val(data.FULLNAME);

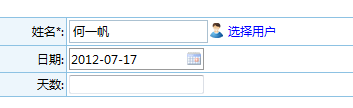
});

};

</script>

至于data数据中的数据结构，可以在预览的时候查看是什么结构。





通过这种方式，用户可以将任意表的数据定制为对话框，然后应用到表单中来。