# Kendoui使用

## 搭建

在header.html中进行所需的css样式和js样式，其中这些样式在配套的文件下面：

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <title></title>  <meta http-equiv=*"Content-Type"* content=*"text/html; charset=utf-8"*/>  <meta http-equiv=*"X-UA-Compatible"* content=*"IE=edge,chrome=1"*>  <meta name=*"\_csrf"* content=*"${\_csrf.token}"*/>  <meta name=*"\_csrf\_header"* content=*"X-CSRF-TOKEN"* />  <link href=*"lib/kendoui/styles/examples-offline.css"* rel=*"stylesheet"*>  <link href=*"lib/kendoui/styles/kendo.common.min.css"* rel=*"stylesheet"*>  <link href=*"lib/kendoui/styles/kendo.rtl.min.css"* rel=*"stylesheet"*>  <link href=*"lib/kendoui/styles/kendo.default.min.css"* rel=*"stylesheet"*>  <link href=*"lib/kendoui/styles/kendo.default.mobile.min.css"* rel=*"stylesheet"*>  <link href=*"lib/bootstrap-3.3.7/css/bootstrap.min.css"* rel=*"stylesheet"*>  <link href=*"lib/font-awesome-4.6.3/css/font-awesome.min.css"* rel=*"stylesheet"*>  <link href=*"lib/kendoui/styles/kendo.common-bootstrap.min.css"* rel=*"stylesheet"* type=*"text/css"*>  <link href=*"lib/kendoui/styles/kendo.bootstrap.min.css"* rel=*"stylesheet"* type=*"text/css"*>  <script src=*"lib/kendoui/js/jquery.min.js"*></script>  <script src=*"lib/kendoui/js/console.js"*></script>  <script src=*"lib/kendoui/js/jszip.min.js"*></script>  <script src=*"lib/kendoui/js/kendo.all.min.js"*></script>  <script src=*"lib/kendoui/js/page.js"*></script>  <script src=*"lib/kendoui/js/dialog.js"*></script>  <script src=*"lib/kendoui/js/kendo.hap.js"*></script>  </head> |

## 2.treeList的使用

treeList用在grid中时，注意在数据库中存数据时，一定要注意一级分类的parentId一定要使用varchar类型，其父类的parentId一定要为null，否则在前端用的时候将查询不到值

例子如下：

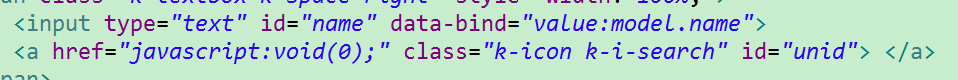
|  |
| --- |
| <#include "../../include/header.html">  <script type=*"text/javascript"*>  **var** viewModel = kendo  .observable({  model : {},  queryResource : **function**(e) {  $('#grid').data('kendoGrid').dataSource.page(1);  },  resetForm : **function**(e) {  **var** formData = viewModel.model.toJSON();  **for** ( **var** k **in** formData) {  viewModel.model.set(k, **null**);  }  }  });  </script>  <body>  <div id=*"example"*>  <div id=*"treelist"*></div>  <script type=*"text/javascript"*>  $(document).ready(**function**(){  **var** baseUrl = "tt/item";  $('#treelist').kendoTreeList({  dataSource:{  transport:{  read:{  url:baseUrl + "/queryItemCat",  dataType:"json"  },  parameterMap: **function**(options, operation) {  **if** (operation == "read") {  **return** prepareQueryParameter(viewModel.model.toJSON(),options)  }  }  },  batch : **true**,  schema:{  data : "rows",  model:{  id:"id",  fields:{  id:{nullable: **false**}  }  }  }  },  height:600,  columns:[  {  field:"name",  title:"商品类目",  width:70  }  ]  });  });  </script>  </div>  </body>  </html> |

## 3.自定义类似lov的组件：

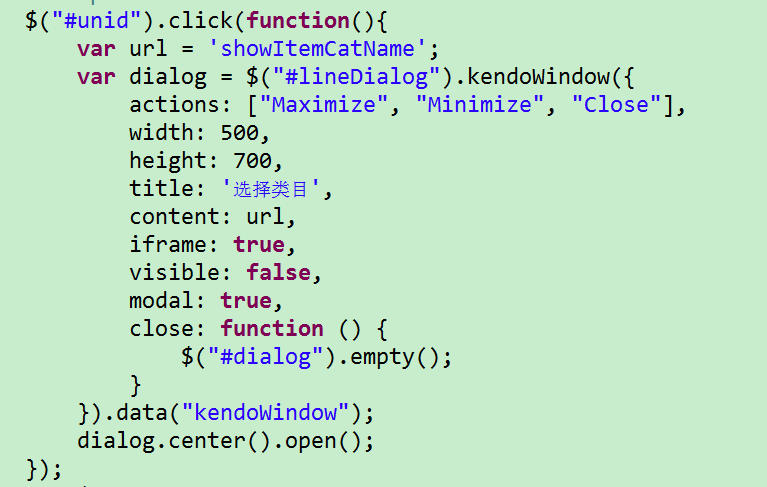
### 3.1 样式如图：



### 3.2 样式代码：



### 3.3 弹出框的代码：

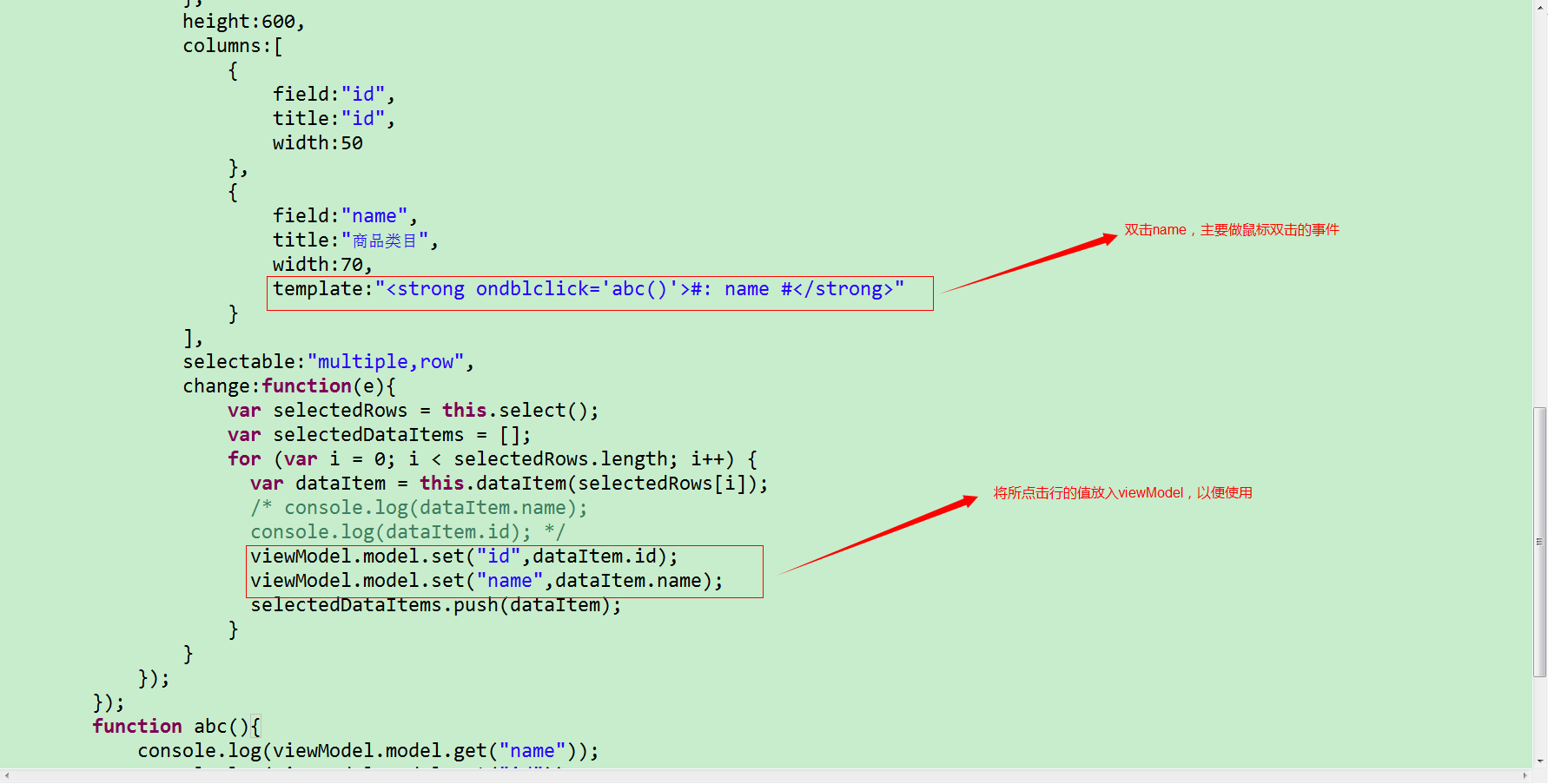


### 3.4 弹出框到另一个页面的代码：

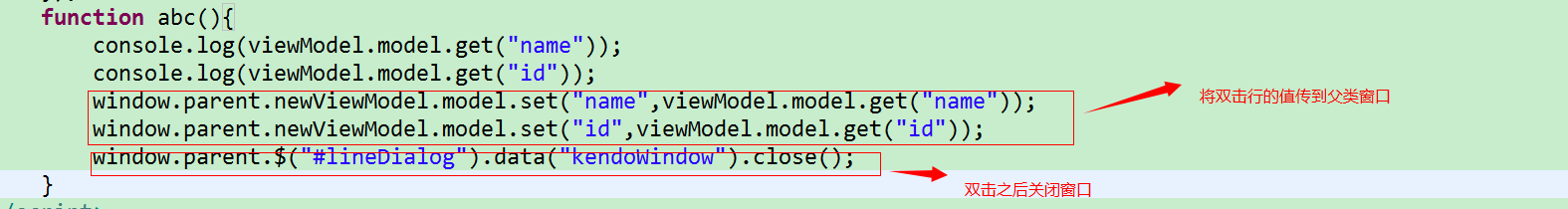
|  |
| --- |
| <#include "../../include/header.html">  <script type=*"text/javascript"*>  **var** viewModel = kendo  .observable({  model : {},  queryResource : **function**(e) {  $('#grid').data('kendoGrid').dataSource.page(1);  },  resetForm : **function**(e) {  **var** formData = viewModel.model.toJSON();  **for** ( **var** k **in** formData) {  viewModel.model.set(k, **null**);  }  }  });  </script>  <body>  <div id=*"example"*>  <div id=*"treelist"*></div>  <script type=*"text/javascript"*>  $(document).ready(**function**(){  **var** baseUrl = "tt/item";  **var** treeList = $('#treelist').kendoTreeList({  dataSource:{  transport:{  read:{  url:baseUrl + "/queryItemCat",  dataType:"json"  },  parameterMap: **function**(options, operation) {  **if** (operation == "read") {  **return** prepareQueryParameter(viewModel.model.toJSON(),options)  }  }  },  batch : **true**,  schema:{  data : "rows",  model:{  id:"id",  fields:{  id:{nullable: **false**}  }  }  }  },  height:600,  columns:[  {  field:"id",  title:"id",  width:50  },  {  field:"name",  title:"商品类目",  width:70,  template:"<strong ondblclick='abc()'>#: name #</strong>"  }  ],  selectable:"multiple,row",  change:**function**(e){  **var** selectedRows = **this**.select();  **var** selectedDataItems = [];  **for** (**var** i = 0; i < selectedRows.length; i++) {  **var** dataItem = **this**.dataItem(selectedRows[i]);  /\* console.log(dataItem.name);  console.log(dataItem.id); \*/  viewModel.model.set("id",dataItem.id);  viewModel.model.set("name",dataItem.name);  selectedDataItems.push(dataItem);  }  }  });  });  **function** abc(){  console.log(viewModel.model.get("name"));  console.log(viewModel.model.get("id"));  window.parent.newViewModel.model.set("name",viewModel.model.get("name"));  window.parent.newViewModel.model.set("id",viewModel.model.get("id"));  window.parent.$("#lineDialog").data("kendoWindow").close();  }  </script>  </div>  </body>  </html> |

### 3.5 注意

1. 其中点击得到所点击行的值，如下图：



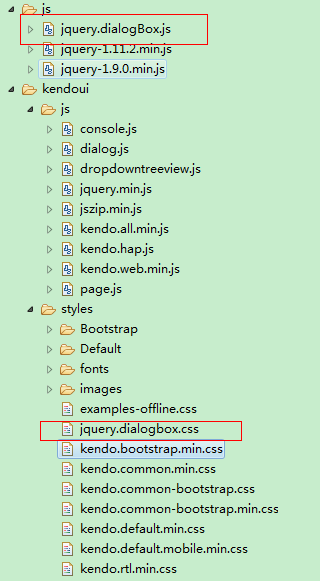
1. 双击执行的方法：



## 弹出提示框

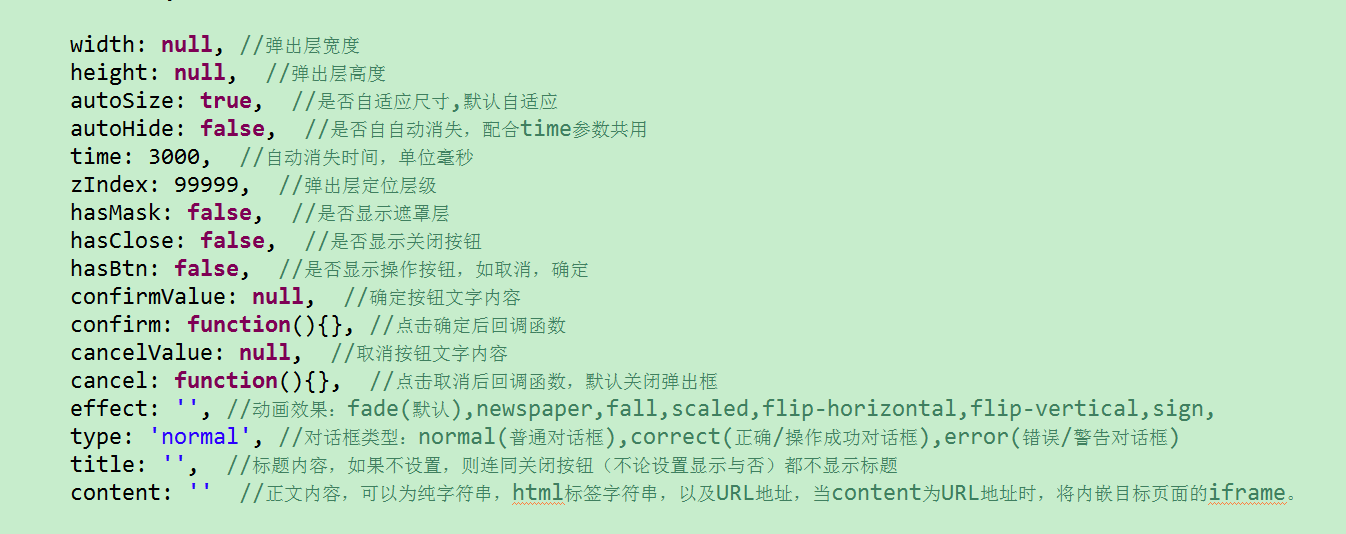
注：因为kendoui的弹出提示框没有找到所以暂时用了jquery原生的弹出提示框插件

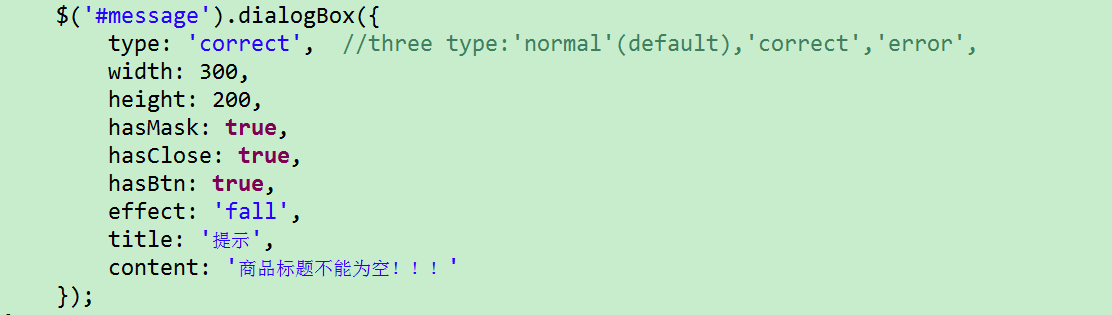
### 4.1 首先导入所需的js和css文件



需要导入上图红色的js和css文件

### 4.2 代码





## 下拉框在grid中的使用（没有动态绑定数据）

### 5.1 首先在页面的头部添加如下代码：

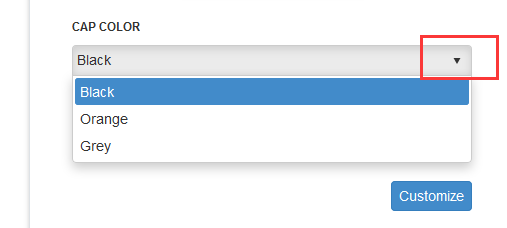
|  |
| --- |
| **var** statusData = [  { text: "正常", value: "1" },  { text: "下架", value: "2" },  { text: "删除", value: "3" }  ]; |

### 5.2 其次在grid中使用如下代码：

|  |
| --- |
| {  field : "status",  title : "状态",  width : "80px",  headerAttributes: {  "class": "table-header-cell",  style: "text-align: center"  },  attributes: {  style: "text-align: center;"  },  template: **function**(dataItem){  **var** v = dataItem.status;  console.log(v);  $.each(statusData,**function**(i,n){  **if**((n.value||'').toLowerCase() == v){  v = n.text||'';  console.log(v);  **return** v;  }  })  **return** v;  },  editor : **function**(container, options){  $('<input name = "' + options.field + '">')  .appendTo(container)  .kendoDropDownList({  dataSource: statusData,  dataTextField: "text",  dataValueField: "value",  valuePrimitive: **true**  });  }  } |

## 6.下拉框在grid中动态的使用

### 6.1 外观如上图所示



### 6.2 from代码声明

|  |
| --- |
| <input class="inputbox" type="text" id="trxDetailId" data-bind="value:model.trxDetailId,  text:model.detailTrxTypeName">  ====================================================================  $("#trxTypeId").kendoDropDownList({  valuePrimitive: true,  optionLabel: " ",//默认选项值  dataTextField: "trxTypeName",//下拉框显示值  dataValueField: "trxTypeId",//下拉框实际值  dataSource: {//数据源  transport: {  read:function(options) {  $.ajax({  type : "POST",  url: "${base.contextPath}/task/taskOrder/TrxType/query",  success: function(json) {  options.success(json.rows);  }  });  }  }  },  select:function(e){//下拉框点击事件  viewModel.model.set("trxDetailId", null);  viewModel.model.set("detailTrxTypeName", null);  },  change : function(e){//下拉框值改变事件  }  }); |

### 6.3 grid 代码声明

|  |
| --- |
| editor: function(container, options){  $('<input required name="' + options.field + '"/>')  .appendTo(container)  .kendoDropDownList({  dataTextField: "meaning",  dataValueField: "lookupCode",  valuePrimitive: true,  dataSource: nodeTypeData,  });  } |

## 快码使用

### 7.1 在页面上添加js



### 7.2 从redis缓存中通过快码的名字查询数据

#### 7.2.1 首先得到redis存的时候的keys

|  |
| --- |
| **public** String getValue(String key){  String string = getFullKey();  String val = jedisClient.hgetValue(string, key);  **return** val;  } |

#### 7.2.2 通过快码的名子可以知道redis的key

|  |
| --- |
| package tt.system.controller;  import java.util.Collections;  import java.util.List;  import java.util.Map;  import javax.servlet.http.HttpServletRequest;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.web.bind.annotation.PathVariable;  import org.springframework.web.bind.annotation.RequestMapping;  import org.springframework.web.bind.annotation.RequestParam;  import org.springframework.web.bind.annotation.ResponseBody;  import org.springframework.web.bind.annotation.RestController;  import com.fasterxml.jackson.core.JsonProcessingException;  import com.fasterxml.jackson.databind.ObjectMapper;  import tt.system.cache.CodeCache;  import tt.system.dto.Code;  import tt.system.dto.CodeValue;  import tt.system.util.JsonUtils;  /\*\*  \*  \* @name CommonController  \* @description 通用的快码查询  \* @author xupf@sunshinepaper.com.cn  \* @version 1.0  \* @date 2017年5月2日  \*/  @RestController  public class CommonController {  @Autowired  private CodeCache codeCache;    @Autowired  private ObjectMapper objectMapper;    @RequestMapping(value = {"/common/{resource}"},produces = {"application/javascript;charset=utf8"})  @ResponseBody  public String getCommonData(@PathVariable String resource, @RequestParam Map<String, String> params, HttpServletRequest request){  StringBuilder sb = new StringBuilder();  params.forEach((k, v) -> {  String string = codeCache.getValue(v);  Code code = JsonUtils.jsonToPojo(string, Code.class);  List<CodeValue> codeValue = getEnabledCodeValues(code);  if(codeValue == null){  try {  this.toJson(sb, k, Collections.EMPTY\_LIST);  } catch (JsonProcessingException e) {  e.printStackTrace();  }  }else{  try {  this.toJson(sb, k, codeValue);  } catch (JsonProcessingException e) {  e.printStackTrace();  }  }  });  return sb.toString();  }    private List<CodeValue> getEnabledCodeValues(Code code){  return code.getCodeValue();  }    protected void toJson(StringBuilder sb, String var, Object data) throws JsonProcessingException {  boolean hasVar = var != null && var.length() > 0;  if(hasVar) {  sb.append("var ").append(var).append('=');  }  sb.append(this.objectMapper.writeValueAsString(data));  if(hasVar) {  sb.append(';');  }  }  } |

#### 7.2.3 通过以上两点可以得到一个code对象，然而在code对象中存有codevalue的集合，进而可以得到code的value和meaning（含义）

#### 7.2.4 之后将6.3中的dataSource改为7.1中的itemStatusData

|  |
| --- |
| editor : **function**(container, options){  $('<input name = "' + options.field + '">')  .appendTo(container)  .kendoDropDownList({  dataSource: itemStatusData,  dataTextField: "meaning",  dataValueField: "value",  valuePrimitive: **true** |

## 图片上传

### 8.1 前端实现

|  |
| --- |
| <form id=*"upload-icon-form"* action=*""* target=*"\_uploadTarget"* enctype=*"multipart/form-data"* method=*"POST"*>  <a href=*"javascript:void(0);"* class=*"file"* >选择文件  <input type=*"file"* name=*"favicon"* id=*"favicon"* onChange=*"preivew(this,'favicon-img');"*/>  </a>  <button type=*"submit"* class=*"btn btn-success submitButton"* onclick="$('#favicon-img').css('visibility','hidden');" disabled=*"disabled"*>上传</button>  <small class=*"help-block"*>  只能上传.png,.jpg,.jpeg,.gif等后缀的图片 注意：上传后需要点击保存  </small>  </form> |

注：enctype="multipart/form-data"为必加项，而且method方法也必须是post

### 8.2 后端实现

# 富文本编辑器

## 9.1 下载方式

参考https://www.kancloud.cn/wangfupeng/wangeditor2/113969上面的文档

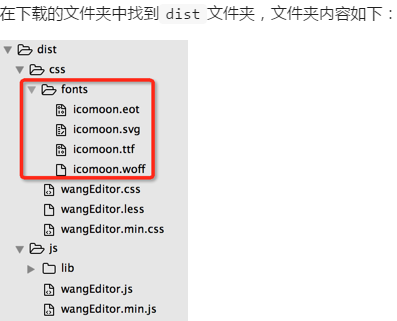
点击 <https://github.com/wangfupeng1988/wangEditor/releases> 下载最新版

使用git下载： git clone https://github.com/wangfupeng1988/wangEditor.git

使用npm安装: npm install wangeditor （注意 wangeditor 全部是小写字母）

使用bower下载：bower install wangEditor （前提保证电脑已安装了bower）

## 9.2 文件结构



注：其中文档结构已经在相同的文件夹下找寻

## 9.3 创建页面

|  |
| --- |
| <link rel=*"stylesheet"* type=*"text/css"* href=*"${base.contextPath}/lib/dist/css/wangEditor.min.css"*>  <script type=*"text/javascript"* src=*"${base.contextPath}/lib/dist/js/wangEditor.min.js"*></script> |

注：在页面顶部加入

## 9.4 创建页面

|  |
| --- |
| <div class=*"row"*>  <div class=*"form-group"*>  <label class=*"col-xs-2 control-label"*>描述</label>  <div class=*"col-xs-5"*>  <textarea id=*"guideContent"*  name=*"guideContent"*  data-bind=*"value:model.guideContent"* rows=*"14"* cols=*"30"*>  </textarea>  <script>  kendo.bind($('#guideContent'), newViewModel);  **var** editor = **new** wangEditor('guideContent');  // 仅仅想移除某几个菜单，例如想移除『插入代码』和『全屏』菜单：  // 其中的 wangEditor.config.menus 可获取默认情况下的菜单配置  editor.config.menus = $.map(wangEditor.config.menus, **function**(item, key) {  **if** (item === 'source') {  **return** **null**;  }  **return** item;  });  editor.config.uploadImgUrl = '${base.contextPath}/fnd/image/publicContentUpload?${\_csrf.parameterName}=${\_csrf.token}';  editor.config.uploadImgFns.onload = **function** (resultText, xhr) {  // resultText 服务器端返回的text  // xhr 是 xmlHttpRequest 对象，IE8、9中不支持  resultText=JSON.parse(resultText)  // 上传图片时，已经将图片的名字存在 editor.uploadImgOriginalName  console.log(resultText.url);  **if**(resultText.code=='S'){  **var** originalName = editor.uploadImgOriginalName || '';  // 如果 resultText 是图片的url地址，可以这样插入图片：  editor.command(**null**, 'insertHtml', '<img src="' + resultText.url + '" alt="' + originalName + '" style="max-width:100%;"/>');  }**else**{  kendo.ui.showErrorDialog({  title:$l('hap.prompt'),  message:resultText.msg  })  }  };  editor.create();  </script>  </div>  </div>  </div> |