**kendo UI速查手册**

本文档基于Kendo UI 2015.1.318版本进行说明。

# 一、基本概念

## 1. 如何开始使用Kendo UI？

1. 首先下载Kendo UI到本地计算机，例如从官方网站<http://www.telerik.com/kendo-ui>或者从国内<http://www.kendoui.io>处下载；

2. 将js和styles文件夹拷贝到项目中；

3. 在页面引入相应的脚本文件和样式文件，包括：

（1）/styles/kendo.common.min.css

（2）/styles/kendo.default.min.css

（3）/js/jquery.min.js（这是1.9.1版本，也可以自行下载基础Jquery库）

（4）/js/kendo.all.min.js

以上即完成使用Kendo UI的准备工作，以后就是Kendo具体的组件或对象使用。

# 二、DataSource

## 1. 什么是DataSource？

顾名思义，这是“数据源”。在Kendo UI中，它被设计为构建所有web Applcation的核心应用，它是本地数据（例如Javascript的数组或对象）或者远程数据（例如web Service返回的JSON、JSONP、OData或者是XML）的抽象表示。

它可以做以下的事情：

（1）从远程终端检索数据；

（2）维护数据的结构和类型（data schema）；

（3）在和远程终端交互的过程中处理数据的序列化（读取或写入）；

（4）在和远程终端交互的过程中进行数据同步（创建、更新、删除）；

（5）维护一个用于远程更新的本地的数据缓存；

（6）计算和维护数据的聚合，排序以及分页；

（7）为数据过滤提供查询语法。

概括起来说，DataSource提供对数据的CRUD操作，以及无论是本地还是远程数据的排序、分页、检索（过滤）、分组和聚合操作。

## 2. 如何将本地数据绑定到DataSource？

下面的代码演示了如何将Javascript的数组指定到DataSource实例的data配置项：

|  |
| --- |
| var movies = [{  title: "Star Wars: A New Hope",  year: 1977  }, {  title: "Star Wars: The Empire Strikes Back",  year: 1980  }, {  title: "Star Wars: Return of the Jedi",  year: 1983  }];  var localDataSource = new kendo.date.DataSource({  data: movies  }); |

## 3. 如何将远程数据绑定到DataSource？

仅仅从获取（读取）数据的角度来说，可以使用DataSource.transport.read来指定一个远程的数据源。例如下面的代码：

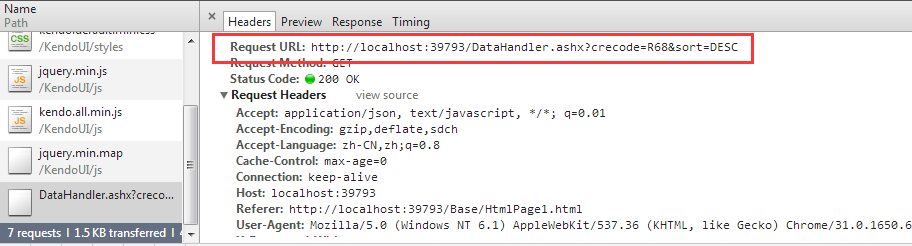
|  |
| --- |
| var remoteDataSource = new kendo.data.DataSource({  transport: {  read: {  url: "../DataHandler.ashx",  type: "get",  dataType: "json",  data: {  crecode: "R68",  sort: "DESC"  }  }  }  }); |

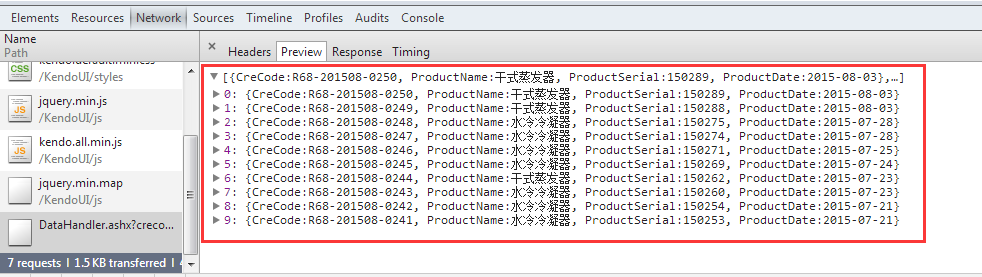
url：指定一个远程的地址来获取程序，这里只针对.NET的ashx进行演示；

type：http请求的类型，get或者是post；

dataType：指定返回的数据格式，通常是json；

data：http请求附加的参数。





## 4. 如何将DataSource应用到widgets上？

大部分Kendo UI的widgets都支持数据绑定，通过对组件的dataSource配置项赋值来将DataSource应用到组件上。有两种方式：

（1）在指定dataSource属性时，对DataSource进行相应的配置，如下：

|  |
| --- |
| $("#chart").kendoChart({  …,  dataSource: new kendo.data.DataSource({  data: [  {  employee: "Joe Smith",  sales: 2000  },  {  employee: "Jane Smith",  sales: 2250  },  {  employee: "Will Roberts",  sales: 1550  }]  }),  …  }); |

（2）声明独立的DataSource实例，将其指定到组件的dataSource配置项上。这样做的好处在于，数据可以在多个widgets之间进行共享或者传递。如下：

|  |
| --- |
| var sharedDataSource = new kendo.data.DataSource({  transport: {  read: {  url: "data-service.json",  dataType: "json"  }  }  });  // 将相同的DataSource绑定到两个组件上面：Chart和Grid  $("#chart").kendoChart({  …,  dataSource: sharedDataSource,  series: [{  field: "sales",  name: "Sales in Units"  }],  categoryAxis: {  field: "employee"  }  });  $("#grid").kendoGrid({  dataSource: sharedDataSource,  …  }); |

## 5. 如何延迟加载数据？

当给DataSource指定了数据以后，实际上，此时仅仅是做了“初始化”的工作。数据并没有被真正加载到DataSource之中。必须通过调用read()方法来加载数据。例如下面的代码：

|  |
| --- |
| var localDataSource = new kendo.data.DataSource({ data: movies });  localDataSource.read(); |

对于指定到widgets上的DataSource，默认就已经执行了read()的动作，因此可以直接在组件上看到数据绑定的效果。当然，如果想不自动绑定显示数据（例如页面初始化的时候出于性能的考虑），可以重载autoBind配置项，将其设置为false就可以避免数据的自动加载。

# 三、Templates

## 1. 什么是Templates？

出于程序开发的需要，我们需要根据获取的数据动态的构建HTML代码段。比如构建一个菜单，以标签<li>来呈现，菜单是存储在数据库里面的，不同的权限看到的菜单项也不一致。此时只有通过ajax来获取菜单数据，然后在Javascript代码里面拼接一个HTML的字符串，将其输出到<ul>的html()里面去。这谓之“传统的HTML string building in JavaScript”。

Kendo UI提供了一个易于使用的，高性能的JavaScript模板引擎，它可以自动将数据嵌合在HTML块中。Kendo UI Templates着眼于针对通用的UI提供一个基础的模板功能，在性能和功能之间更着重前者的需求。

## 2. Templates渲染界面的语法有几种？

有三种语法：

（1）#=……#：以HTML的语法来渲染UI；

（2）#: ……#：忽略HTML语法，直接将获取得值渲染到UI上面；

（3）# …… #：在界面上引用Javascript代码段。

对上面的第（1）种和第（2）种情况，他们的区别是：假设有一个数据源是下面的格式：

|  |
| --- |
| var data = { firstName: "<b>Keanu</b>" }; |

那么下面是第一种模板语法：

|  |
| --- |
| <div id="example"></div>  <script>  var template = kendo.template("#= firstName #");  var result = templete(data);  $('#example').html(result);  </script> |

此时，界面输出的是粗体的keanu，即**keanu**。因为它是按照HTML语法将<b>解释为粗体，并将其渲染到id为example的div里面。

第二种模板语法：

|  |
| --- |
| <div id="example"></div>  <script>  var template = kendo.template("#： firstName #");  var result = templete(data);  $('#example').html(result);  </script> |

此时，界面输出的是<b>keanu<b>。它直接将值渲染到界面UI上面。

## 3. 如何快速建立Templates？

通过指定kendo.template内的字符串来创建一个template为我们提供了很大的便利。但是，这样做很不直观，依然是一种“HTML string building in JavaScript”。可以通过以下的方式来便利创建。首先，创建一个<script></script>代码块，在这里代码块里面包含需要模板的HTML预制内容。如下，注意type属性的设置：

|  |
| --- |
| <script id="javascriptTemplate" type="text/x-kendo-template">  <input type="text" value="#: firstName #" />  </script> |

其次，将这个id为javascriptTemplate的html()指定给模板。如下：

|  |
| --- |
| <div id="example"></div>  <script>  var template = kendo.template($('#javascriptTemplate').html());  var data = { firstName: "Keanu" };  var result = template(data);  $('#example').html(result);  </script> |

有一点需要注意，写在script里面的html代码是没有自动提示的。因此，可以现在页面的其他地方写好需要的HTML代码，然后拷贝到<script>代码块里面。

## 4. 什么时候使用#...#语法？

如果在界面上需要动态的，重复性的进行模板构建，可以使用这个语法。比如，前面说到动态构建菜单，假设从数据库里面获取了5个菜单项，都需要放置在标签<li>里面，那么我们无需在模板中写五次，只要做一个循环就可以了，如下：

|  |
| --- |
| <div id="example"></div>  <script id="javascriptTemplate" type="text/x-kendo-template">  <ul>  # for(var i=0; i<data.length; i++) { #  <li>#: data[i] #</li>  # } #  </ul>  </script>  <script>  var template = kendo.template($('#javascriptTemplate').html());  var data = ["keanu", "diago", "ada", "sike", "sixin"];  var result = template(data);  $('#example').html(result);  </script> |

总结来说，##语法可以包含任意的js代码段，也适用于用户自定义的变量。或者说，完全可以在页面的某个地方写好完整的javascript代码，然后将其拷贝到模板中。

## 5. 如何在模板中使用字符#？

如果在模板中要输出字符#，则需要转义。格式为“\\#”，如下所示：

|  |
| --- |
| <script type="text/javascript">  var templateString = '<a href="\\#index">#: myName #</a>';  var template = kendo.template(templateString);  $("#example").html(template({ myName: "Todd" }));  </script> |

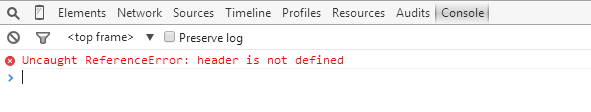
## 6. 如何获得极限性能（模板解析最快速度）？

前面说了，Kendo UI Template更加注重性能。不同于其他的一些模板引擎，它只提供了3种模板语法，而没有开发更多的模板构建表达式。在Template中，有一个配置项叫做useWithBlock，这个配置项默认值为true。在这个模式下，Template的性能（或者说速度）并不是最快的，但是有一个好处，它可以直接使用传入数据的键值。如下代码所示：

|  |
| --- |
| <div id="example"></div>  <script type="text/x-kendo-template" id="temp">  <div>  <h1 class='header'>#= header #</h1>  <h2 class='header2'>#= header2 #</h2>  <ul>  # for(var i = 0, l = list.length; i < l; i++){#  <li class='item'>#= list[i] #</li>  #}#  </ul>  </div>  </script>  <script>  var template = kendo.template($('#temp').html());  var myDemoData = {  header: "Header",  header2: "Header2",  list: ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10']  };  var result = template(myDemoData);  $('#example').html(result);  </script> |

可以看到，我们在模板中直接使用了“header”、“header2”和“list[i]”来获取传入数据的相应的值。如果此时我们把useWithBlock设置为false，再查看运行效果，则程序会报一个“Uncaught ReferenceError: header is not defined”的错误。如下：

|  |
| --- |
| var template = kendo.template($('#temp').html(), { useWithBlock: false }); |



为了修复这个错误，我们修改模板代码，将数据的header包含进来，如下：

|  |
| --- |
| <script type="text/x-kendo-template" id="temp">  <div>  <h1 class='header'>#= data.header #</h1>  <h2 class='header2'>#= data.header2 #</h2>  <ul>  # for(var i = 0, l = data.list.length; i < l; i++){#  <li class='item'>#= data.list[i] #</li>  #}#  </ul>  </div>  </script> |

因此，总结来说，关于useWithBlock选项。如果设置为true，影响性能，但是利于开发书写模板。如果设置为false，性能最优，但是需要注意模板的书写格式。这种模式适用于移动终端。

## 7. 如何复用模板？

在构建大型项目往往要用到很多模板，此时会面临一个问题：我们在什么地方定义和维护模板？Kendo UI提供了两种模板定义方式：

* 本地模板：将模板定义在本身的HTML文件中
* 远程模板：将模板定义在外部的文件中

# 四、Grid

## 1. 如何绑定本地数据？

使用dataSource.data属性和columns属性。其中dataSource.data指定数据源，columns指定表格的列结构。代码如下：

|  |
| --- |
| <script>  var products = [{  ProductID: 1,  ProductName: "分汽缸",  Serial: "15S1-1",  FinishDate: "2015-07-23",  Price: 1200,  Weight: 200  }, {  ProductID: 2,  ProductName: "储气罐",  Serial: "15C1-5",  FinishDate: "2015-07-24",  Price: 800,  Weight: 50  }, {  ProductID: 3,  ProductName: "吸附器",  Serial: "2015R-01-45",  FinishDate: "2015-07-24",  Price: 4000,  Weight: 300  }];  $(document).ready(function () {  $('#grid').kendoGrid({  dataSource: {  data: products  },  columns: [  { field: "ProductName", title: "产品名称" },  { field: "Serial", title: "产品编号", width: "150px" },  { field: "FinishDate", title: "制造日期", width: "150px" },  { field: "Price", title: "价格", format: "{0:c}", width: "100px" }  ]  });  });  </script> |



## 2. 如何绑定服务器端数据？

具体参阅：[1.3如何使用DataSource获取远程数据？](#_3._如何使用DataSource获取远程数据？)，代码如下：

|  |
| --- |
| <script>  var remoteDataSource = new kendo.data.DataSource({  transport: {  read: {  url: "../DataHandler.ashx",  type: "get",  dataType: "json",  data: {  crecode: "R68",  sort: "DESC"  }  }  }  });  $(document).ready(function () {  $('#grid').kendoGrid({  dataSource: remoteDataSource,  columns: [  { field: "CreCode", title: "证书编号" },  { field: "ProductName", title: "产品名称", width: "150px" },  { field: "ProductSerial", title: "产品编号", width: "150px" },  { field: "ProductDate", title: "制造日期", width: "100px" }  ]  });  });  </script> |

从后台返回的数据要拼凑成一个格式规范的JSON字符串，将其写回前端。如下：

|  |
| --- |
| StringBuilder sb = new StringBuilder();  sb.Append("[");  foreach (DataRow dr in dt.Rows)  {  sb.Append("{");  sb.Append("\"CreCode\":").Append("\"").Append(dr["CreCode"].ToString()).Append("\", ");  sb.Append("\"ProductName\":").Append("\"").Append(dr["ProductName"].ToString()).Append("\", ");  sb.Append("\"ProductSerial\":").Append("\"").Append(dr["ProductSerial"].ToString()).Append("\", ");  sb.Append("\"ProductDate\":").Append("\"").Append(DateTime.Parse(dr["ProductDate"].ToString()).ToString("yyyy-MM-dd")).Append("\"");  sb.Append("},");  }  string temp = sb.ToString();  string result = temp.Substring(0, temp.Length - 1) + "]"; |



# 五、AutoComplete

## 1. 如何从数据库获取数据绑定到AutoComplete？

通过DataSource来配置数据，指定到AutoComplete即可。如下：

|  |
| --- |
| <script>  var countries = ["中国", "俄罗斯", "美国", "英国", "法国", "日本", "印度", "澳大利亚", "德国"];  $('#countries').kendoAutoComplete({  dataSource: countries  });  </script> |

同理，只要远程终端将数据组织成类似的格式并返回，就可以绑定显示在AutoComplete当中了。如下代码所示：

|  |
| --- |
| <script>  $('#countries').kendoAutoComplete({  dataSource: new kendo.data.DataSource({  transport: {  read: {  url: "Handlers/AutoCompleteHandler.ashx",  type: "get",  dataType: "json",  data: {  controlType: "BasicRemoteBound"  }  }  }  })  });  </script> |
| if (controlType == "BasicRemoteBound")  {  DataTable dt = SqlHelper.ExecuteDataset(CommandType.Text, "SELECT Name FROM Production.Product").Tables[0];  foreach (DataRow dr in dt.Rows)  {  sb.Append("\"").Append(dr["Name"].ToString()).Append("\"");  sb.Append(",");  }  string temp = sb.ToString();  string result = "[" + temp.Substring(0, temp.Length - 1) + "]";  context.Response.Write(result);  } |

## 2. 如何获取被选中的值？

首先是获取autocomplete的实例，然后通过内置的value()方法来获取被选中的值。参阅如下代码：

|  |
| --- |
| $("#btnGetValue").click(function () {  var autocomplete = $("#countries").data("kendoAutoComplete");  alert(autocomplete.value());  }); |

## 3. 能够预先指定一个值在AutoComplete里面吗？

可以，同样是使用value()方法来设定值。如下：

|  |
| --- |
| var autocomplete = $("#countries").data("kendoAutoComplete");  autocomplete.value("中国"); |

## 4. 能绑定键值对吗？

很多情况下，出于界面友好的角度，我们在弹出的提示框里面显示的直观显式的文本。但是为了数据库操作的便利，我们需要的又是这个文本所对应的键值。例如存在Products表有ID和Name两个字段。我们在AutoComplete中输入提示的肯定是Name信息，但是在选择Name之后我们需要却是对应的ID的值，以便于后期的数据库CRUD操作。

AutoComplete并不支持这样的键值对的提示选择，也就是说，通过value()方法，只能取到选择的文本值。如果想要获取value对应的key值，可以考虑ComboBox或者是DropDownList组件。

从DataSource本身来说，并不限定绑定在AutoComplete上面的数据必须是一维数组，完全可以返回符合JSON要求的键值对。这个时候，可以通过dataTextField属性来指定需要显示的字段。如下：

|  |
| --- |
| var data = [  { id: 1, name: "Apples" },  { id: 2, name: "Oranges" }  ];  $("#autocomplete").kendoAutoComplete({  dataTextField: "name",  dataSource: data  }); |

## 5. 什么情况下使用AutoComplete组件？

符合下列情况下使用AutoComplete组件较为合适：

（1）只需要文本值的时候——因为AutoComplete只能取到文本值；

（2）总的提示项数量不多——因为操作者的无法记住太多的内容；

（3）或者虽然数量多，但是内容具有明显的可以排查追溯的轨迹。

因此，如果总提示数量不多，可以将数据建立在本地以减轻服务器压力。如果从数据库获取远程数据的话，也可以考虑建立本地数据副本，缓存数据。

十五、关于DataSource的高级话题