2016-10-17

李文强、周鹏

湖南心莱信息科技有限公司

Magicodes.Echarts使用教程

目录

[1 概要 2](#_Toc464427085)

[2 Magicodes.ECharts工作原理 3](#_Toc464427086)

[2.1 架构说明 3](#_Toc464427087)

[2.1.1 Axis 4](#_Toc464427088)

[2.1.2 CommonDefinitions 4](#_Toc464427089)

[2.1.3 Components 4](#_Toc464427090)

[2.1.4 JsonConverter 4](#_Toc464427091)

[2.2 Series 6](#_Toc464427092)

[2.3 ValueTypes 6](#_Toc464427093)

[2.4 EChartsOptions 7](#_Toc464427094)

[2.5 TimelineOptions 8](#_Toc464427095)

[3 Magicodes.EchartsJs 8](#_Toc464427096)

[3.1 全局配置 8](#_Toc464427097)

[3.2 组件配置 8](#_Toc464427098)

[3.3 Demo 9](#_Toc464427099)

[4 5分钟上手Magicodes.EChart 9](#_Toc464427100)

[4.1 获取EChart 9](#_Toc464427101)

[4.2 引用Magicodes.ECharts 9](#_Toc464427102)

[4.3 引用Echart相关脚本 10](#_Toc464427103)

[4.4 编写第一个图表 11](#_Toc464427104)

[4.4.1 编写控制器代码 11](#_Toc464427105)

[4.4.2 编写前端代码 12](#_Toc464427106)

[4.4.3 效果 12](#_Toc464427107)

[4.5 编写动态图表 13](#_Toc464427108)

[5 相关示例 13](#_Toc464427109)

博客使用Word发博，发布后，排版会出现很多问题，一一修正工作量极大，敬请谅解。可加群获取原始文档。

# 概要

Magicodes.ECharts是心莱团队基于百度EChart封装的开源的.NET类库，以便让用户更快更便捷的上手开发EChart图表。本篇主要讲解Magicodes.ECharts的相关使用。

注意：EChart图表插件是由百度提供的一套前端图表库，可以流畅运行在PC端和移动设备上。Echarts具备完善的兼容性和丰富的图表类型，目前已经存在2.0和3.0两个版本，Magicodes.ECharts针对的是最新版本的封装。ECharts的官网为：<http://echarts.baidu.com/> 。

**官方博客：**[**http://www.cnblogs.com/codelove/**](http://www.cnblogs.com/codelove/)

**相关开源库地址：**[**https://github.com/xin-lai**](https://github.com/xin-lai)

在开始之前，您需要了解以下内容：

* **Magicodes.ECharts是基于百度EChart封装的开源的.NET类库，封装的目的便于使用强类型语言约束后台代码，以便于更好维护和重构代码，并且封装自身业务**
* **Magicodes.EChartsJS是基于knockoutJs封装的组件，结合Magicodes.ECharts能够很方便的利用Ajax加载Echart图表**
* **目前Magicodes.ECharts已经支持大部分图表，但是尚不能支持所有图表和所有情形**
* **本着按需设计的原则，在遇到不支持的情形，希望您能够自行扩展并贡献自己的代码。众人拾材火焰高，Magicodes.ECharts在不断地实践中会更加完善的**
* **在设计上，我们不会过多设计，但是后续版本可能会不断地进行重构**

# Magicodes.ECharts工作原理

## 架构说明

Magicodes.ECharts目前主要由以下部分组成：

### Axis

直角坐标系 grid 中的轴的相关配置封装，比如x轴、y轴等。

### CommonDefinitions

通用定义，目前定义了以下内容：

* Align（水平对齐）
* Label（标签）
* Orients（图例列表的布局朝向）
* Symbols（标记的图形）
* TextAlign（标题文本水平对齐）
* TextStyles（文本样式）
* VerticalAlign（垂直对齐）

### Components

组件定义。目前定义了以下内容：

* DataZoom（区域缩放）
  + DataZoomInside（内置型数据区域缩放组件）
  + DataZoomSlider（滑动条型数据区域缩放组件）
* Grid（直角坐标系内绘图网格）
* Legend（图例组件）
* TimeLine（timeline 组件）
* Title（标题）
* ToolBox（工具箱）
* ToolTip（提示框组件）

### JsonConverter

定义了一些自定义实现的JSON转换器，依赖自JSON.NET。

* ValueConverter

ValueConverter用于相关值JSON转换，需要实现IValue<T>接口。如下面例子：

/// <summary>

/// 小数数组

/// </summary>

[JsonConverter(typeof(ValueConverter<double[]>))]

public class ArrayNumberValue : IValue<double[]>, INumberOrArrayNumberValue

{

public double[] Value { get; set; }

}

/// <summary>

/// 数值

/// </summary>

[JsonConverter(typeof(ValueConverter<double>))]

public class NumberValue : ILeftValue, ITopValue, IRightValue, IBottomValue, IValue<double>, INumberOrArrayNumberValue, INumberOrStringValue

{

public NumberValue(double? value = null)

{

if (value.HasValue)

{

Value = value.Value;

}

}

public double Value { get; set; }

}

接下来，在DataZoom组件上用到了我们的定义，如下所示：

public abstract class DataZoom

{

….省略其他代码

/// <summary>

/// 设置 dataZoom-inside 组件控制的 y轴（即yAxis，是直角坐标系中的概念，参见 grid）。

/// 不指定时，当 dataZoom-inside.orient 为 'vertical'时，默认控制和 dataZoom 平行的第一个 yAxis。但是不建议使用默认值，建议显式指定。

/// 如果是 number 表示控制一个轴，如果是 Array 表示控制多个轴。

/// </summary>

public INumberOrArrayNumberValue XAxisIndex { get; set; }

}

在使用时，我们可以这样赋值：

new DataZoomInside()

{

Start=30,

End=70,

XAxisIndex=new ArrayNumberValue() {Value=new double[] { 0,1} }

}

当然也可以给其赋予数值：

new DataZoomInside()

{

Start=30,

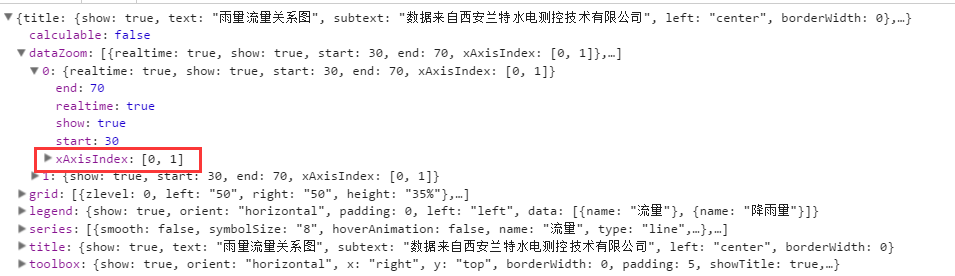
End=70,

//XAxisIndex=new ArrayNumberValue() {Value=new double[] { 0,1} }

XAxisIndex=new NumberValue(1)

}

在生成JSON时，改转换器会生成对应正确的JSON，如下图所示：



## Series

定义了系列。每个系列通过 Type 决定自己的图表类型。

目前定义有：

* BarSeries
* LineSeries
* 其他图表类型大都可以通过实例化基类然后指定系列类型

## ValueTypes

值类型。因EChart的许多属性配置均可以支持不同的数据类型和对象，Magicodes.Echarts特对此进行了封装，以进行约束。目前定义有以下值类型：

* AlignValue
* ArrayNumberValue
* DoubleValue
* IBottomValue
* ILeftValue
* INumberOrArrayNumberValue
* INumberOrStringValue
* IRightValue
* ISymbolValue
* ITopValue
* IValue
* NumberValue
* StringValue
* SymbolValue
* VerticalAlignValue

具体使用请参考本框架源码。

## EChartsOptions

EChart配置基类。允许设置各个组件和对象来定义Echart。

如下面Demo：

[HttpGet]

public JavaScriptJsonResult Demo1()

{

var provinces = new List<object>() { "岳阳", "长沙", "北京", "上海", "武汉", "贵州", "大连", "成都", "深圳" };

var valueList = new List<object>() { 50, 100, 130, 39, 31, 66, 11, 34, 9 };

var chartOptions = new EChartsOption

{

Title = new Title("粉丝分布统计图") { Left = new AlignValue(Align.center) },

Series = new Series.Series[]

{

new BarSeries

{

Name = "粉丝分布",

Data = valueList,

MarkPoint = new MarkPoint

{

Data = new List<MarkData>

{

new MarkData {Type = MarkPointDataTypes.max, Name = "最大值"},

new MarkData {Type = MarkPointDataTypes.min, Name = "最小值"}

}

},

MarkLine = new MarkLine

{

Data = new List<MarkData>

{

new MarkData {Type = MarkPointDataTypes.average, Name = "平均值"}

}

}

}

},

XAxis = new XAxis[1] { new XAxis { Type = AxisTypes.category, Data = provinces } },

YAxis = new YAxis[1] { new YAxis { Type = AxisTypes.value } }

};

return this.ToEChartResult(chartOptions);

}

## TimelineOptions

Timeline配置基类。允许设置各个组件和对象来定义Echart。

# Magicodes.EchartsJs

Magicodes.EChartsJs为针对Echart封装的Ajax加载的knockoutjs库，需要依赖以下javascript库：

* Jquery
* Knockoutjs

关于Magicodes.EChartsJs，你可以在开源库源码中的项目Magicodes.ECharts.Demo找到。

该组件主要有以下配置：

## 全局配置

基于window.mcs.echarts.settings进行配置，比如设置全局主题：

mcs.echarts.settings.theme="macarons"

## 组件配置

* ajaxUrl:Ajax加载路径
* isBlockUI:是否显示加载遮罩层
* theme:主题

## Demo

<div data-bind='component:{name: "echart",params: { ajaxUrl: "/ChartDemo/Demo1" ,theme:"macarons"}}' style="height: 280px; width: 100%"></div>

<div data-bind='component:{name: "echart",params: { ajaxUrl: "/ChartDemo/Demo2" ,theme:"macarons"}}' style="height: 280px; width: 100%"></div>

<div data-bind='component:{name: "echart",params: { ajaxUrl: "/ChartDemo/Demo3" ,theme:"macarons"}}' style="height: 800px;width: 100%"></div>

# 5分钟上手Magicodes.EChart

## 获取EChart

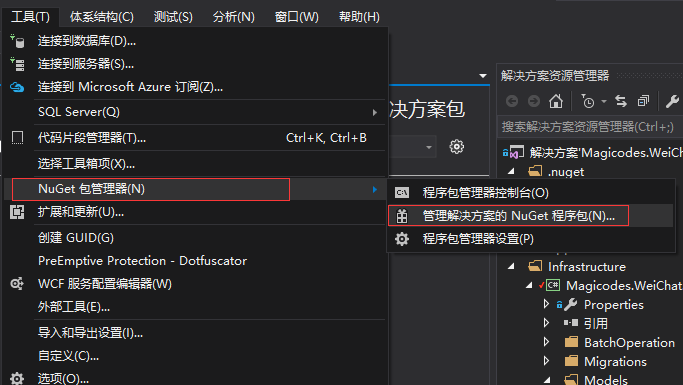
你可以在此下载EChart：<http://echarts.baidu.com/download.html>

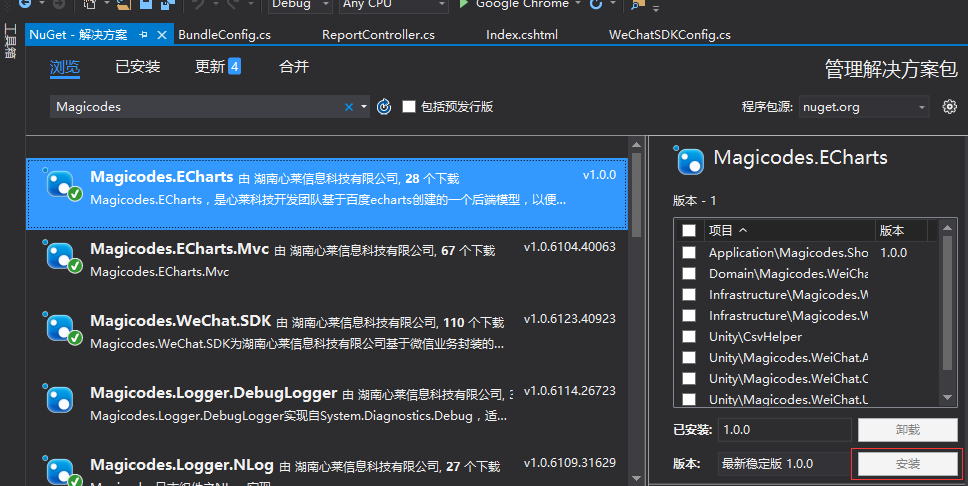
## 引用Magicodes.ECharts

使用Nuget包下载，请分别下载：

* Magicodes.ECharts
* Magicodes.ECharts.Mvc
* Magicodes.EChartsJs

相关操作如下图所示：





也可以在Github下载源代码引用，地址为：<https://github.com/xin-lai/Magicodes.ECharts>

## 引用Echart相关脚本

推荐使用ASP.NET Bundle技术，配置如下所示：

//echarts

bundles.Add(new ScriptBundle("~/plugins/echarts").Include(

"~/Scripts/plugins/echart/echarts.js",

"~/Scripts/plugins/echart/theme/macarons.js",

"~/Scripts/components/magicodes.echart.js"));

注意：如果使用到了相关主题，请注意引用主题脚本。

那么在View页引用如下：

@Scripts.Render("~/bundles/jquery")

@Scripts.Render("~/bundles/knockout")

@Scripts.Render("~/bundles/bootstrap")

@Scripts.Render("~/plugins/echarts")

其中，Jquery、Knockoutjs、Echarts脚本是必须的，其他请按需引用。

## 编写第一个图表

### 编写控制器代码

首先我们需要编写控制器代码，以便Ajax调用。如下所示：

[HttpGet]

public JavaScriptJsonResult Demo2()

{

var data = new List<object>() {

new {value=400, name= "搜索引擎"},

new {value=335, name="直接访问"},

new {value=310, name="邮件营销"},

new {value=274, name="联盟广告"},

new {value=235, name="视频广告"}

};

var chartOptions = new EChartsOption

{

Title = new Title("搜索来源") { Left = new AlignValue(Align.center) },

Series = new Series.Series[]

{

new Series.Series

{

Name = "访问来源'",

Data = data,

Type=SeriesTypes.pie,

Radius="55%",

}

}

};

return this.ToEChartResult(chartOptions);

}

请注意引用相关命名空间，并注意，ToEChartResult为扩展方法，用于将上述配置对象输出为JSON序列化内容。

### 编写前端代码

前端组件代码如下：

<div data-bind='component:{name: "echart",params: { ajaxUrl: "/ChartDemo/Demo2" ,theme:"macarons"}}' style="height: 280px; width: 100%"></div>

注意，定义好Echart组件之后，我们还需要在页面上添加以下代码：

@section Scripts

{

<script>

$(function () {

ko.applyBindings({});

});

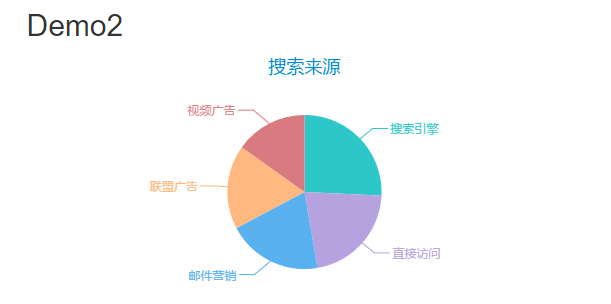
</script>

}

该脚本用于绑定KO组件，请务必添加。

### 效果

至此，代码编写完毕，效果如下所示：



## 编写动态图表

有时候，我们希望图表能够即时的进行数据刷新并动态变化，那么使用Magicodes.EChartsJs就比较简单了，如下面Demo：

<script>

$(function () {

var viewModel = function () {

var self = this;

this.demo4Url = ko.observable('/ChartDemo/Demo1');

this.init = function () {

//设置定时器

setInterval(function () {

//可以通过Url传参

self.demo4Url(self.demo4Url() == '/ChartDemo/Demo1' ? '/ChartDemo/Demo2' : '/ChartDemo/Demo1');

}, 2000);

};

self.init();

};

ko.applyBindings(new viewModel());

});

</script>

HTML：

<div class="row">

<div class="col-md-12">

<h2>Demo4——动态图</h2>

<div data-bind='component:{name: "echart",params: { ajaxUrl: demo4Url ,theme:"macarons"}}' style="height: 400px;width: 100%"></div>

</div>

</div>

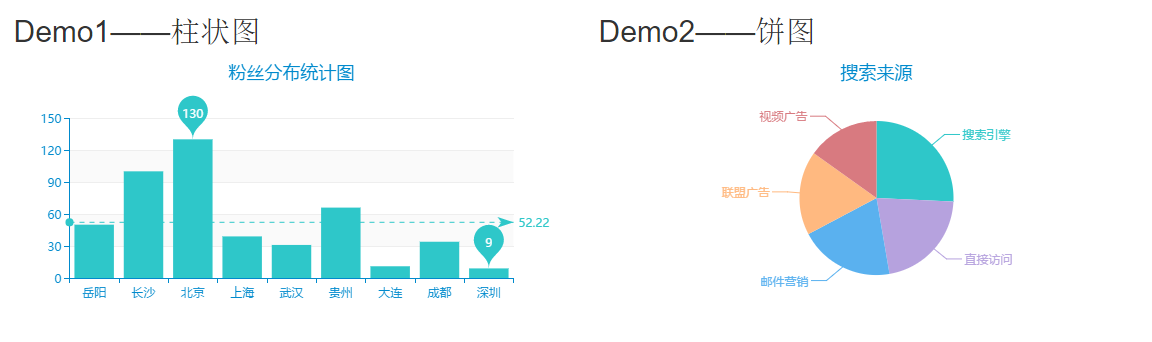
**上述Demo实现了图表每隔两秒的刷新，会自动从Demo1切换为Demo2的图表，在实际应用中，您可以可以通过Url传参或者按照自身业务来动态加载图表数据。**

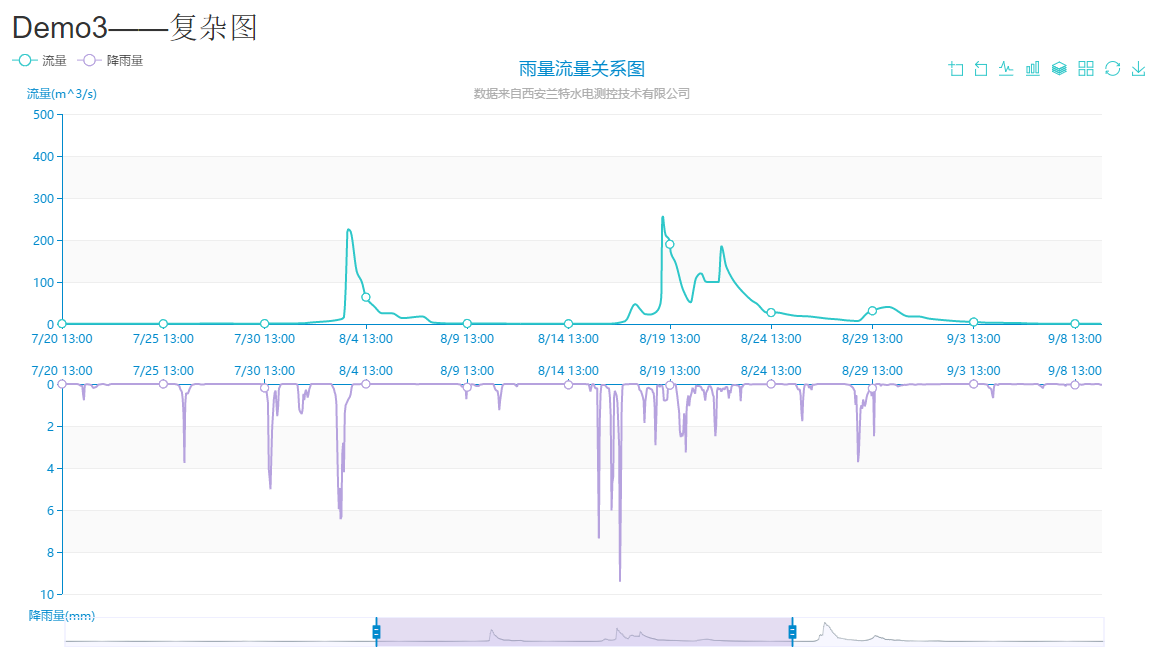
**由上面代码得知，Magicodes.EChartsJs组件的ajaxUrl参数不仅仅支持字符串，还支持绑定ko的监视器，如果图表需要即时刷新，只需要给监视器赋值就可以了，当值产生变化时，图表会自动刷新，如下面代码：**

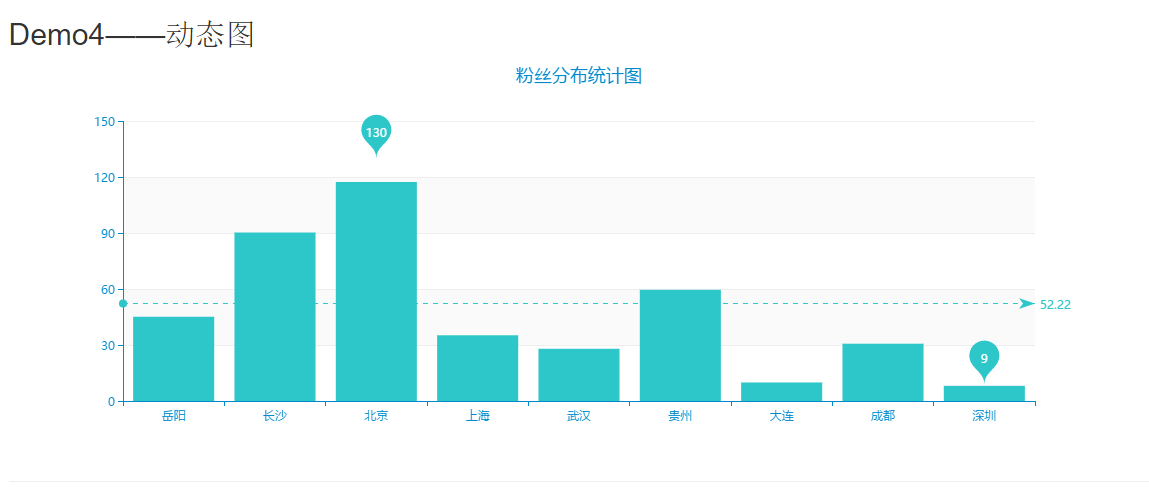
self.demo4Url(self.demo4Url() == '/ChartDemo/Demo1' ? '/ChartDemo/Demo2' : '/ChartDemo/Demo1');

# 相关示例

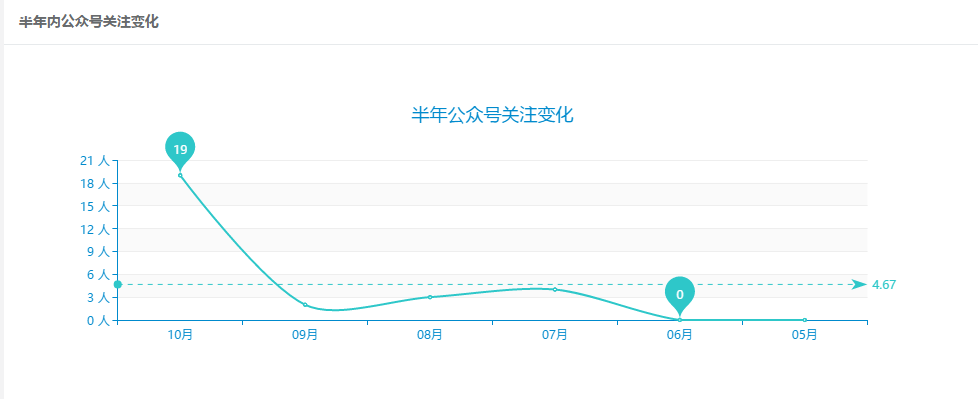
目前在开源库中，定义了4个Demo，如下图所示：

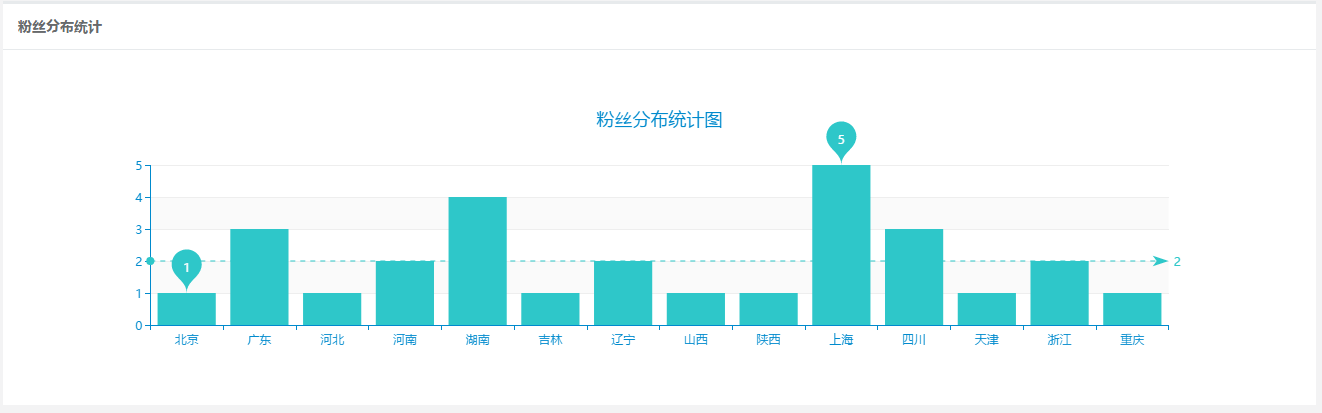


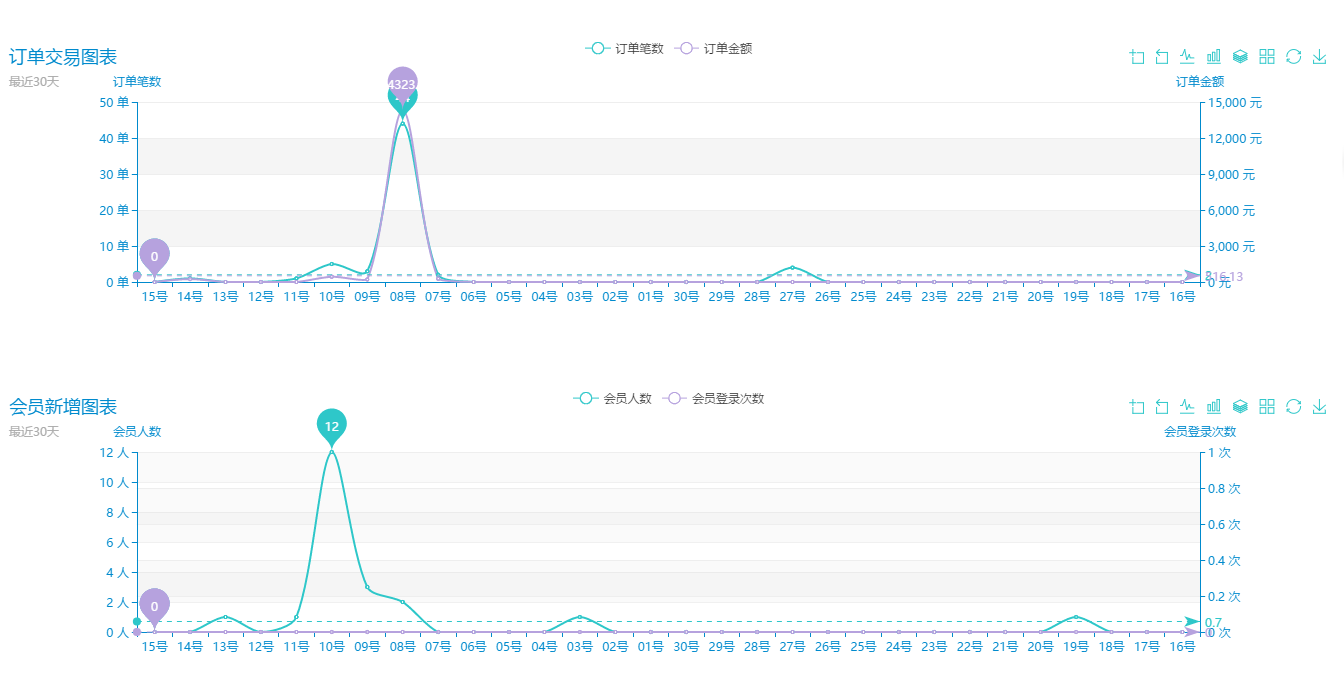
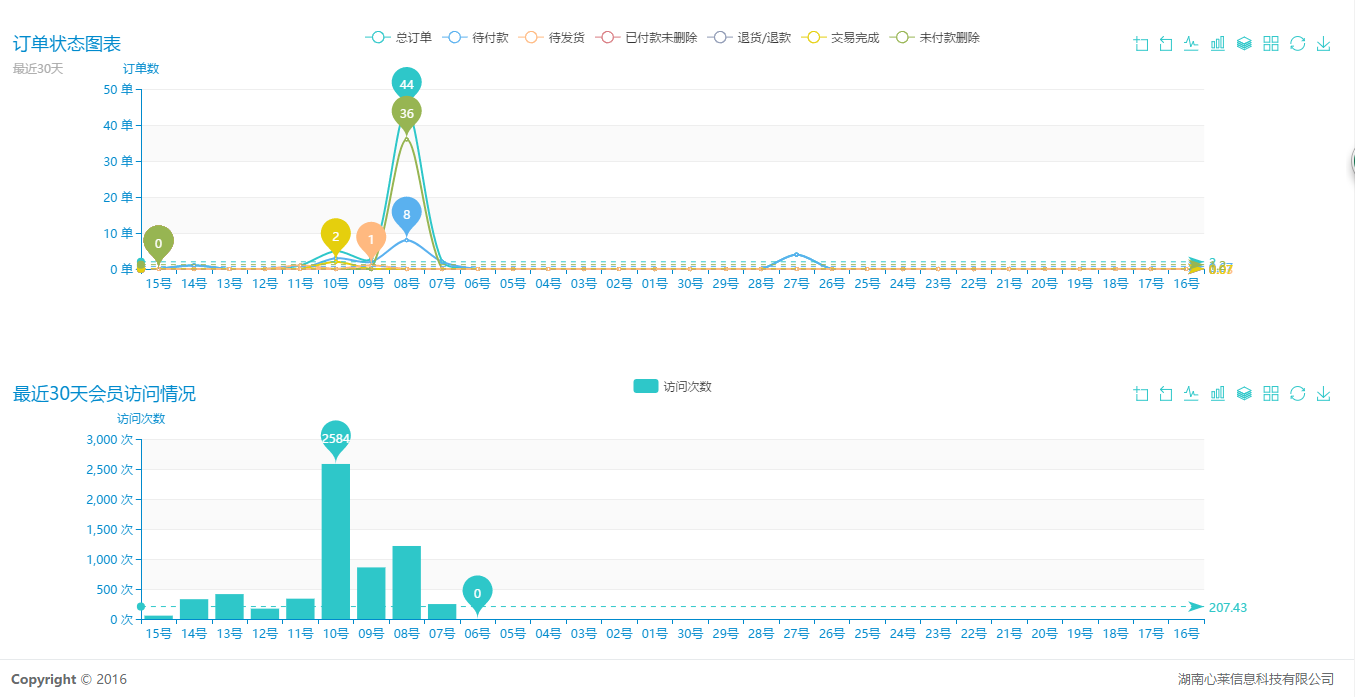




下面是我们使用该组件在实践中的一些示例：





目前Magicodes.ECharts尚不能支持所有图表和所有情形，如果你对本项目有兴趣，可以贡献自己的代码哦。**本着按需设计的原则，Magicodes.ECharts在不断地实践中会更加完善的。**