

Echarts

Designer

```
//SvcPlugin_demo_demoEchartsdemo
vsPluginComponentModule.factory('$vcPlugin_demo_demoEcharts',
['$vsPluginRegister', '$timeout', function ($vsPluginRegister, $timeout) {

    //echartsoption
    var optionTemplate = {
        title: {
            show: false
        },
        legend: {
            show: false,
            orient : 'horizontal',
            x : 'right',
            y : 'top',
            data:[],
            textStyle: {
                color: "#999",
                fontSize: 12
            }
        },
        grid: {
            x: 42,
            y: 34,
            x2: 11,
            y2: 25,
            borderWidth: 0,
            borderColor: "#f0f0f0"
        },
        tooltip : {
            show:true,
            confine: true,
            trigger: 'axis',
            axisPointer: {
                type: 'shadow'
            },
            position: function (pos, params, dom, rect, size) {
                var obj = {top: 0};
                obj['left'] = pos[0] < size.viewSize[0] / 2 ? pos[1] :
pos[1]-size.contentSize[0];
                return obj;
            }
        },
        calculable : false,
```

```
confine: true,
animationDuration: 500,
xAxis : [{
  type : 'category',
  boundaryGap: true,
  axisLine: {
    lineStyle: {
      color: '#DDD',
      width: 1,
      shadowBlur: 0,
      shadowOffsetX: 0,
      shadowOffsetY: 0
    },
    show: true
  },
  splitArea: {
    show: false
  },
  splitLine: {
    lineStyle: {
      color: '#F6F6F6'
    }
  },
  axisTick: {
    show: false
  },
  axisLabel: {
    textStyle:{
      color: '#666',
      fontSize: 12
    }
  },
  data : ['', '', '', '', '', '', '']
}],
yAxis : [{
  name: "",
  type : 'value',
  axisLine: {
    lineStyle: {
      color: '#B0B0B0',
      width: 1,
      shadowBlur: 0,
      shadowOffsetX: 0,
      shadowOffsetY: 0
    },
    show: true
  },
  splitArea: {
    show: false
  },
  splitLine: {
    lineStyle: {
      color: '#F6F6F6'
    }
  }
}]
```

```

        }
    },
    axisTick: {
        show: false
    },
    axisLabel: {
        textStyle: {
            color: '#666',
            fontSize: 12
        }
    }
}],
series: [{
    name: "",
    itemStyle: {
        normal: {
            color: "#006BB7",
            lineStyle : {
                color: "#006BB7"
            }
        }
    },
    data: [820, 932, 901, 934, 1290, 1330, 1320],
    type: 'bar'
}]
};

```

```

var factory = {

    //[]
    showDataCategory: true,
    //[]
    showBorderCategory: true,
    //[]
    showBasicCategory: true,
    //[]
    showFixedCategory: true,
    //[]
    showEventCategory: true,
    //[]
    showTitleCategory: true,
    //[]
    showThresholdCategory: true,

    /* */
    init: function(scope, element, component, $compile){
        scope.element = element;
        scope.component = component;
        //
        component.config.pageFilter = false;
        //
        scope.component.config.chartDimensionCount = 1;
    }
};

```

```

        //1
        scope.component.config.minDimensionCount = 1;
        //1
        scope.component.config.minMeasureCount = 1;
        //
        if(element.height() == 0){
        $timeout(function(){
            factory.init(scope, element, component, $compile, $sce, $timeout);
        },100);
        }else{
            factory.initChart(scope, element, component, $compile);
        }

        },

        initChart: function(scope, element, component, $compile){
        var myChart = echarts.init(element[0]);
        var option = component.config.chartConfig;
        if(option == null){
            //echartsoption
            option = optionTemplate;
            option.noDataLoadingOption = {};
            //optionchartConfig
            component.config.chartConfig = option;
            component.config.legendOrient = "horizontal";
        }

        option.tooltip.position = function (pos, params, dom, rect,
size) {
            var obj = {top: 0};
            obj['left'] = pos[0] < size.viewSize[0] / 2 ? pos[0] :
pos[0]-size.contentSize[0];
            return obj;
        };

        //echartssetOption
        try{
            myChart.setOption(option);
        }catch(e){
        }

        //chartcontext
        component.context.chart = myChart;
        //echarts
        component.context.resize = function(){
            $timeout(function(){
                if(component.context.chart.resize){
                    component.context.chart.resize();
                }
            }, 100);
        }

    },

    /*

```

```

        *
        */
        buildDataDescription: function(dataDescription, scope, element,
component, $compile){
            //
            scope.$on(event_refreshComponentData, function(target, param){
                scope.queryComponentData(param, {
                    onSuccess: function(){
                        refreshChartView(scope, element, component,
$compile);
                    }
                });
            });

            //
            scope.$on(event_chartDimensionValueChange, function(s, event){
                var dataIndex = -1;
                var axisLabels = component.config.chartConfig.xAxis[0].data;
                for(var i = 0; i < axisLabels.length; i++){
                    if(""+axisLabels[i] === ""+event.source.value){
                        dataIndex = i;
                        break;
                    }
                }
                if(dataIndex < 0){
                    component.context.chart.dispatchAction({
                        type: 'hideTip'
                    });
                    return;
                }
                if(dataIndex > -1){
                    component.context.chart.dispatchAction({
                        type: 'showTip',
                        dataIndex: dataIndex,
                        seriesIndex: 0
                    });
                }
            });
        },

        /*
        *
        */
        buildChartDescription: function(scope, element, component,
$compile){
            //[]
            buildChartDescription(scope, element, component, $compile);
            //[]
            buildAxisDescription(scope, element, component, $compile);
            //[]
            buildMeasureDescription(scope, element, component, $compile);
        }
    };

```

```

    //[]
    var buildAxisDescription = function(scope, element, component,
$compile){
        var category = {
            name: "axis",
            title: "",
            groups: []
        };
        component.description.categories.push(category);
    }

    //[]
    var buildMeasureDescription = function(scope, element, component,
$compile){
        var category = {
            name: "measures",
            title: "",
            groups: []
        };
        component.description.categories.push(category);
    }

    //[]
    var buildChartDescription = function(scope, element, component,
$compile){
        var category = {
            name: "chart",
            title: "",
            groups: []
        };
        component.description.categories.push(category);

        //
category.groups.push({
    title: {
        text: ""
    },
    elements: [{
        title: "",
        type: "radio",
        bind: "showLegend",
        items: [{
            name: "",
            value: "show"
        }],
        name: "",
        value: "hide"
    }], {
        title: "",
        type: "radio-icon",
        bind: "legendOrient",

```

```

items: [{
  name: "",
  value: "vertical",
  icon: "fa fa-ellipsis-v"
},{
  name: "",
  value: "horizontal",
  icon: "fa fa-ellipsis-h"
}]
}, {
  title: "",
  type: "radio-icon",
  bind: "legendPosY",
  items: [{
    name: "",
    value: "top",
    icon: "fa fa-minus",
    pStyle: "padding-top:0;padding-bottom:12px;"
  },{
    name: "",
    value: "center",
    icon: "fa fa-minus",
    pStyle: "padding-top:6px;padding-bottom:6px;"
  },{
    name: "",
    value: "bottom",
    icon: "fa fa-minus",
    pStyle: "padding-top:12px;padding-bottom:0px;"
  }],
  show: function(ele){
    return scope.component.config.legendOrient != null &&
scope.component.config.legendOrient === "vertical";
  }
}, {
  title: "",
  type: "radio-icon",
  bind: "legendPosX",
  items: [{
    name: "",
    value: "left",
    icon: "fa fa-align-left"
  },{
    name: "",
    value: "center",
    icon: "fa fa-align-center"
  },{
    name: "",
    value: "right",
    icon: "fa fa-align-right"
  }],
  show: function(ele){
    return scope.component.config.legendOrient != null &&
scope.component.config.legendOrient === "horizontal";
  }
}

```

```

    }
  }]
});

//showLegend
scope.$watch('component.config.showLegend', function(newValue, oldValue){
  if(newValue != null && (oldValue == null || oldValue !== newValue)){
    var option = component.config.chartConfig;
    if(option.legend == null){
      option.legend = {
        show: false,
        orient : 'horizontal',
        x : 'right',
        y : 'top',
        data:[""],
        textStyle: {
          color: "#999",
          fontSize: 12
        }
      };
      option.legend.show = newValue === "show";
      refreshChartView();
    }else{
      option.legend.show = newValue === "show";
      scope.component.context.chart.setOption(option, true);
    }
  }
});

}

//
var refreshChartView = function(scope, element, component, $compile){
  //
  var dimensions = component.config.datasourceConfig.dimensions;
  //
  var measures = component.config.datasourceConfig.measures;
  //echartsoption
  var option = component.config.chartConfig;
  //
  var data = component.context.data;
  if(data == null){
    return;
  }

  //
  if(option.series.length > measures.length){
    var newSeries = [];
    for(var i = 0; i < measures.length; i++){
      newSeries.push(option.series[i]);
    }
    option.series = newSeries;
  }
}

```



```

//X
var xAxisLabels = [];
for(var i = 0; i < data.length; i++){
    xAxisLabels.push(data[i][dimensions[dimensions.length-1].name]);
}
if(xAxisLabels.length == 0){
    xAxisLabels = [""];
}
//
var legendData = [];
for(var m = 0; m < measures.length; m++){
    var serieData = [];
    for(var i = 0; i < data.length; i++){
        if(isNaN(data[i][measures[m].name])){
            data[i][measures[m].name] = null;
        }
        serieData.push(data[i][measures[m].name]);
    }
    if(serieData.length == 0){
        serieData = [""];
    }
    if(option.series[m] == null){
        var newOption = $.extend(true, {}, optionTemplate);
        option.series[m] = newOption.series[0];
    }
    option.series[m].data = serieData;
    option.series[m].name = measures[m].label;

    //
    if(component.config["measureAlias_"+m] != null){
        option.yAxis[0].name = component.config["measureAlias_"+m];
    }
    if(option.yAxis[0].name == null || option.yAxis[0].name.length == 0){
        option.yAxis[0].name = measures[m].label;
    }

    //
    if(component.config["columnColor_"+m] == null){
        option.series[m].itemStyle.normal.color = "#4c87b5";
    }else{
option.series[m].itemStyle.normal.color=component.config["columnColor_"+m];
    }
    //
    option.series[m].stack = component.config["stack_"+m];
    if(VSUtils.isEmpty(option.series[m].stack) || !option.series[m].stack){
        delete option.series[m].stack;
    }

    //
    var legendValue = measures[m].label;
    if(component.config["measureAlias_"+m] != null &&

```

```

component.config["measureAlias_"+m].length > 0){
    option.series[m].name = component.config["measureAlias_"+m];
    legendValue = component.config["measureAlias_"+m];
}
legendData.push(legendValue);

        //seriesname
if(option.series[m].name == null || option.series[m].name.length == 0){
    option.series[m].name = measures[m].label;
}
}

    option.xAxis[0].data = xAxisLabels;
    if(option.legend != null){
        option.legend.data = legendData;
    }

        //1Y
if(measures.length > 1 || option.legend.show === true){
    option.yAxis[0].name="";
}

setTimeout(function(){
    component.context.chart.dispatchAction({
        type: 'hideTip'
    });
    setTimeout(function(){
        component.context.chart.setOption(option, true);
    });
});

};

// "demo"

```

```
$vsPluginRegister.register("demo", "demoEcharts", factory);  
}]);
```

View

```
var build_demoEcharts_component = function(scope, element, $compile,  
$timeout, $sce){  
  
    var component = scope.component;  
  
    var myChart = echarts.init(element[0]);  
    var option = component.config.chartConfig;  
    if(option == null){  
        //echartsoption  
        option = optionTemplate;  
        option.noDataLoadingOption = {};  
        //optionchartConfig  
        component.config.chartConfig = option;  
        component.config.legendOrient = "horizontal";  
    }  
  
    option.tooltip.position = function (pos, params, dom, rect, size) {  
        var obj = {top: 0};  
        obj['left'] = pos[0] < size.viewSize[0] / 2 ? pos[0] :  
pos[0]-size.contentSize[0];  
        return obj;  
    };  
  
    //echartssetOption  
    try{  
        myChart.setOption(option);  
    }catch(e){  
    }  
  
    //chartcontext  
    component.context.chart = myChart;  
    //echarts  
    component.context.resize = function(){  
        $timeout(function(){  
            if(component.context.chart.resize){  
                component.context.chart.resize();  
            }  
        }, 100);  
    }  
}
```

```

//designer.js
var refreshChartView = function(){
    //
    var dimensions = component.config.datasourceConfig.dimensions;
    //
    var measures = component.config.datasourceConfig.measures;
    //echartsoption
    var option = component.config.chartConfig;
    //
    var data = component.context.data;
    if(data == null){
return;
}

//
    if(option.series.length > measures.length){
        var newSeries = [];
        for(var i = 0; i < measures.length; i++){
            newSeries.push(option.series[i]);
        }
        option.series = newSeries;
    }

    //X
    var xAxisLabels = [];
    for(var i = 0; i < data.length; i++){
        xAxisLabels.push(data[i][dimensions[dimensions.length-1].name]);
    }
    if(xAxisLabels.length == 0){
        xAxisLabels = [""];
    }
    //
    var legendData = [];
    for(var m = 0; m < measures.length; m++){
        var serieData = [];
        for(var i = 0; i < data.length; i++){
            if(isNaN(data[i][measures[m].name])){
                data[i][measures[m].name] = null;
            }
            serieData.push(data[i][measures[m].name]);
        }
        if(serieData.length == 0){
            serieData = [""];
        }
        if(option.series[m] == null){
            var newOption = $.extend(true, {}, optionTemplate);
            option.series[m] = newOption.series[0];
        }
        option.series[m].data = serieData;
        option.series[m].name = measures[m].label;

        //

```

```

        if(component.config["measureAlias_"+m] != null){
            option.yAxis[0].name = component.config["measureAlias_"+m];
        }
        if(option.yAxis[0].name == null || option.yAxis[0].name.length == 0){
            option.yAxis[0].name = measures[m].label;
        }

        //
        if(component.config["columnColor_"+m] == null){
            option.series[m].itemStyle.normal.color = "#4c87b5";
        }else{

option.series[m].itemStyle.normal.color=component.config["columnColor_"+m];
        }
        //
        option.series[m].stack = component.config["stack_"+m];
        if(VSUtils.isEmpty(option.series[m].stack) || !option.series[m].stack){
            delete option.series[m].stack;
        }

        //
        var legendValue = measures[m].label;
        if(component.config["measureAlias_"+m] != null &&
component.config["measureAlias_"+m].length > 0){
            option.series[m].name = component.config["measureAlias_"+m];
            legendValue = component.config["measureAlias_"+m];
        }
        legendData.push(legendValue);

        //seriesname
        if(option.series[m].name == null || option.series[m].name.length == 0){
            option.series[m].name = measures[m].label;
        }
    }

    component.context.originalXAxisLabels = null;
    if(!VSUtils.isEmpty(component.config.xAxisLabelScript)){
        component.context.originalXAxisLabels = angular.copy(xAxisLabels);
        try{
            var f = eval("(function(labelData){
"+Base64.decode(component.config.xAxisLabelScript)+"})");
            f.call(null, xAxisLabels);
        }catch(e){
            console.log(e);
        }
    }

    option.xAxis[0].data = xAxisLabels;
    if(option.legend != null){
        option.legend.data = legendData;
    }

    //1Y
    if(measures.length > 1 || option.legend.show === true){

```

```

    option.yAxis[0].name="";
}

setTimeout(function(){
    component.context.chart.dispatchAction({
        type: 'hideTip'
    });
    setTimeout(function(){
        component.context.chart.setOption(option, true);
    });
});
};

//
scope.$on(event_refreshComponentData, function(target, param){
    scope.queryComponentData(param, {
        onSuccess: function(){
            refreshChartView(scope, element, component, $compile);
        }
    });
});

//designer.js
scope.$on(event_chartDimensionValueChange, function(s, event){
    var dataIndex = -1;
var axisLabels = component.config.chartConfig.xAxis[0].data;
for(var i = 0; i < axisLabels.length; i++){
    if(""+axisLabels[i] === ""+event.source.value){
        dataIndex = i;
        break;
    }
}
if(dataIndex < 0){
    component.context.chart.dispatchAction({
        type: 'hideTip'
    });
    return;
}
if(dataIndex > -1){
    component.context.chart.dispatchAction({
        type: 'showTip',
        dataIndex:dataIndex,
        seriesIndex: 0
    });
}
}

```

```
});
```

```
}
```