Add a chart

09/10/2016 • 4 minutos para ler • Colaboradores tudo

Neste artigo

This page demonstrates how you can add charts to your extensions. Charts can be added to any Azure DevOps Services extension.

These charts are easy to create, resizable, interactive and consistent with the Azure DevOps Services look and feel. The following chart types are supported:

1. Line
2. Bar
3. Column
4. Area
5. Stacked bar
6. Stacked column
7. Stacked area
8. Pie
9. Pivot table
10. Histogram

> If you're in a hurry and want to get your hands on the code right away, you can download the complete samples. Once downloaded, go to the `charts` folder, then follow the packaging and publishing instructions to publish the sample extension. The extension contains sample chart widgets.

How to organize your code

For the purposes of this tutorial, we'll be creating a widget and adding a chart to it. To do so, in the `home` folder for your project, create a `chart.html` file with the following contents:

HTML file

HTML Copiar

```html
<!DOCTYPE html>
<html>
<head>
    <script src="sdk/scripts/VSS.SDK.js"></script>
    <script src="scripts/pie-chart.js"></script>
</head>
<body>
    <div class="widget">
        <h2 class="title">Chart Widget</h2>
        <div id="Chart-Container"></div>
    </div>
</body>
</html>
```

> Use the **npm install** command to retrieve the SDK: `npm install vss-web-extension-sdk`. To learn more about the SDK, visit the Client SDK GitHub Page.

Ensure that the `VSS.SDK.js` file is inside the `sdk/scripts` folder so that the path is `home/sdk/scripts/VSS.SDK.js`.

Images

Add the following images to an `img` folder in your project directory so that the path is `home/img/logo.png` and `home/img/CatalogIcon.png`:

1. Extension logo
2. Catalog icon

Extension manifest file

In the `home` folder of your project, create your extension manifest file. Create a `vss-extension.json` file with the following contents:

JSON Copiar

```json
{
    "manifestVersion": 1,
    "id": "samples-charts",
    "version": "1.0.0",
    "name": "Interactive Charts in Azure DevOps Services",
    "description": "Samples of interactive charts from the Chart SDK.",
    "publisher": "Fabrikam",
    "targets": [
        {
            "id": "Microsoft.VisualStudio.Services"
        }
    ],
    "icons": {
        "default": "img/logo.png"
    },
    "demands": ["contribution/ms.vss-dashboards-web.widget-sdk-version-2", "contribution/ms.vss-
web.charts-service"],
    "contributions": [
        {
            "id": "chart",
            "type": "ms.vss-dashboards-web.widget",
            "targets": [
                "ms.vss-dashboards-web.widget-catalog"
            ],
            "properties": {
                "name": "Sample Chart",
                "description": "A simple chart widget with no configuration and hard-coded
data.",
                "catalogIconUrl": "img/CatalogIcon.png",
                "uri": "chart.html",
                "supportedSizes": [
                    {
                        "rowSpan": 2,
                        "columnSpan": 2
                    }
                ],
                "supportedScopes": [
                    "project_team"
                ]
            }
        }
    ],
    "files": [
        {
            "path": "chart.html",
            "addressable": true
        },
        {
            "path": "sdk/scripts/VSS.SDK.js",
            "addressable": true
        },
        {
            "path": "img",
            "addressable": true
        },
        {
            "path": "scripts",
            "addressable": true
        }
    ],
    "scopes": [
    ]
}
```

Before uploading this extension, you'll need to update the `publisher` to yours.

Put the following code snippets into a `chart.js` file in a `scripts` folder, so that the path is `home/scripts/chart.js`. Then follow the packaging and publishing instructions to publish your extension.

Charts

Pie chart

This sample renders a pie chart. The `data` and `labelValues` have been hardcoded here, and would need to be changed to the data you want to visualize.

JavaScript Copiar

```
VSS.init({
        explicitNotifyLoaded: true,
        usePlatformStyles: true
    });

VSS.require([
        "TFS/Dashboards/WidgetHelpers",
        "Charts/Services"
        ],
        function (WidgetHelpers, Services) {
        WidgetHelpers.IncludeWidgetStyles();
        VSS.register("PieChart", function () {
             return {
             load:function() {
                return Services.ChartsService.getService().then(function(chartService){
                    var $container = $('#Chart-Container');
                    var chartOptions = {
                        "hostOptions": {
                            "height": "290",
                            "width": "300"
                        },
                        "chartType": "pie",
                        "series": [{
                            "data": [11, 4, 3, 1]
                        }],
                        "xAxis": {
                            "labelValues": ["Design", "On Deck", "Completed", "Development"]
                        },
                        "specializedOptions": {
                            "showLabels": "true",
                            "size": 200
                        }
                    };

                    chartService.createChart($container, chartOptions);
                    return WidgetHelpers.WidgetStatusHelper.Success();
                });
                }
            }
          });
    VSS.notifyLoadSucceeded();
});
```

Here, the chart's size is defined in `hostOptions`. The series property is an array and contains a single object with data in it. The `xAxis` object contains `labelValues` which correspond to the `data`. For pie charts, we also have some special options that are defined by the `specializedOptions` property. Here, we're explicitly enabling data labels for the pie chart. We also need to set the size of the pie chart by specifying its outer diameter.

Rendering the chart requires a container to render it in, the chart options, and a call to the Chart Service to get initialize the chart and render it.

Stacked area chart

This sample renders a stacked area chart.This chart type is ideal for comparing a relationship of parts to a whole and highlighting general trends across categories. It is commonly used to compare trends over time. This sample also specifies a custom color for one of the series being rendered.

JavaScript Copiar

```javascript
VSS.init({
        explicitNotifyLoaded: true,
        usePlatformStyles: true
    });

VSS.require([
        "TFS/Dashboards/WidgetHelpers",
        "Charts/Services"
        ],
        function (WidgetHelpers, Services) {
        WidgetHelpers.IncludeWidgetStyles();
        VSS.register("StackedAreaChart", function () {
             return {
             load:function() {
                return Services.ChartsService.getService().then(function(chartService){
                    var $container = $('#Chart-Container');
                    var chartOptions = {
                        "hostOptions": {
                            "height": "290",
                            "width": "300"
                        },
                        "chartType": "stackedArea",
                        "series": [
                            {
                                "name": "Completed",
                                "data": [1,3,4,3,6,1,9,0,8,11]
                            },
                            {
                                "name": "Development",
                                "data": [1,1,0,3,0,2,8,9,2,8]
                            },
                            {
                                "name": "Design",
                                "data": [0,0,0,6,1,1,9,9,1,9],
                                "color": "#207752"
                            },
                            {
                                "name": "On Deck",
                                "data": [1,2,4,5,4,2,1,7,0,6]
                            }
                        ],
                        "xAxis": {
                            "labelFormatMode": "dateTime_DayInMonth",
                            "labelValues": [
                                "1/1/2016",
                                "1/2/2016",
                                "1/3/2016",
                                "1/4/2016",
                                "1/5/2016",
                                "1/6/2016",
                                "1/7/2016",
                                "1/8/2016",
                                "1/9/2016",
                                "1/10/2016"
                            ]
                        }
                    }
                    chartService.createChart($container, chartOptions);
                    return WidgetHelpers.WidgetStatusHelper.Success();
                });
                }
             }
            });
    VSS.notifyLoadSucceeded();
});
```

Pivot table

This sample renders a Pivot Table. This chart type helps arrange and summarize complex data in a tabular form.

JavaScript Copiar

```
VSS.init({
        explicitNotifyLoaded: true,
        usePlatformStyles: true
    });

VSS.require([
        "TFS/Dashboards/WidgetHelpers",
        "Charts/Services"
        ],
        function (WidgetHelpers, Services) {
        WidgetHelpers.IncludeWidgetStyles();
        VSS.register("PivotTable", function () {
             return {
             load:function() {
                return Services.ChartsService.getService().then(function(chartService){
                    var $container = $('#Chart-Container');
                    var chartOptions = {
                        "hostOptions": {
                            "height": "290",
                            "width": "300"
                        },
                        "chartType": "table",
                        "xAxis": {
                            "labelValues": ["Design","In Progress","Resolved","Total"]
                        },
                        "yAxis": {
                            "labelValues": ["P1","P2","P3","Total"]
                        },
                        "series": [
                            {
                                "name": "Design",
                                "data": [
                                    [0,0,1],
                                    [0,1,2],
                                    [0,2,3]
                                ]
                            },
                            {
                                "name": "In Progress",
                                "data": [
                                    [1,0,4],
                                    [1,1,5],
                                    [1,2,6]
                                ]
                            },
                            {
                                "name": "Resolved",
                                "data": [
                                    [2,0,7],
                                    [2,1,8],
                                    [2,2,9]
                                ]
                            },
                            {
                                "name": "Total",
                                "data": [
                                    [3,0,12],
                                    [3,1,15],
                                    [3,2,18],
                                    [0,3,6],
                                    [1,3,15],
                                    [2,3,24],
                                    [3,3,10]
                                ],
                                "color": "rgba(255,255,255,0)"
                            }
                        ]
                    }

                    chartService.createChart($container, chartOptions);
                    return WidgetHelpers.WidgetStatusHelper.Success();
                });
                }
               }
              });
        VSS.notifyLoadSucceeded();
});
```