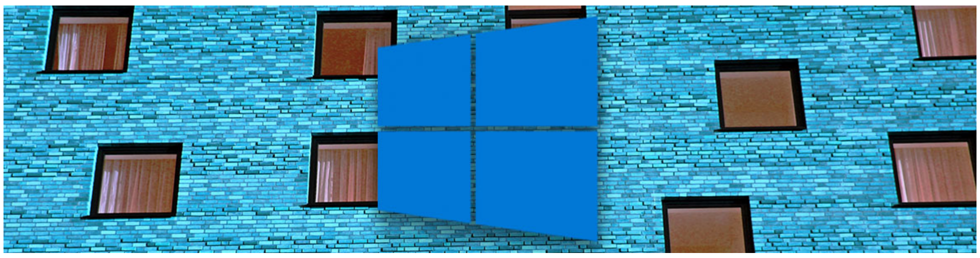
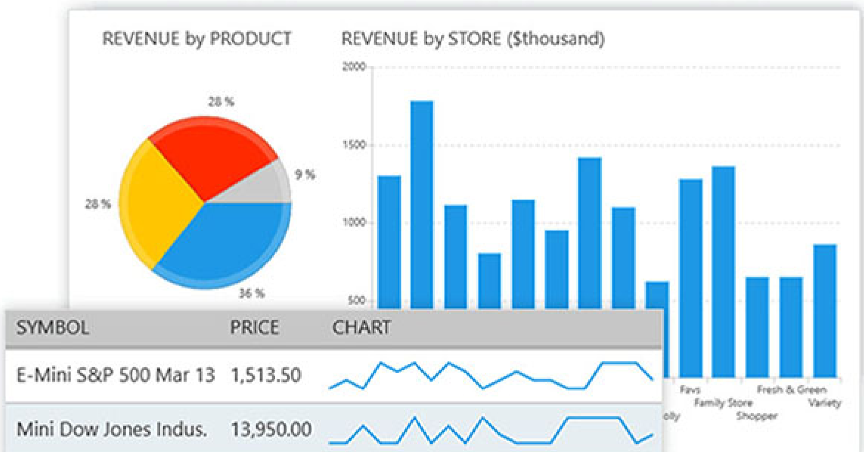
Getting Started with Telerik UI for UWP

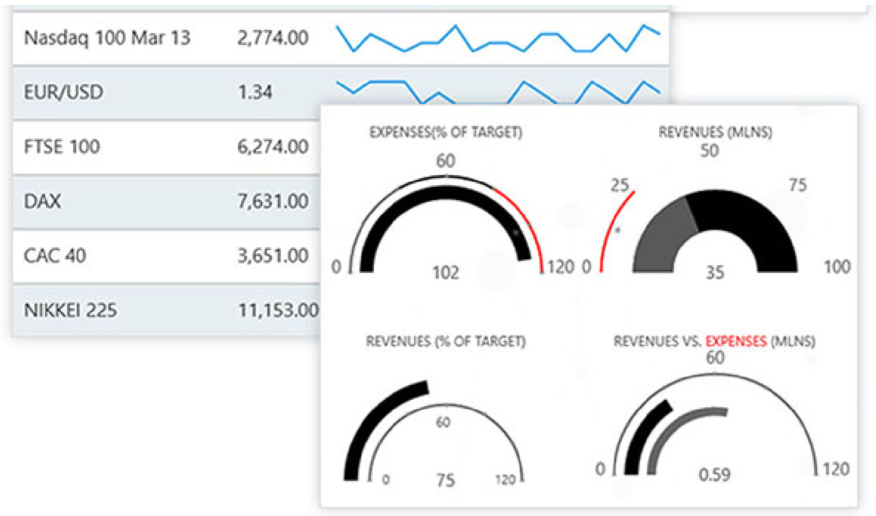
.NET by John Bristowe | August 1, 2017 | 0 Comments



Have you heard of Telerik UI for UWP? It's a suite of UI controls for Universal Windows Platform (UWP). You can find the source code (Apache License v2.0) and the documentation on GitHub. What's more, we're now accepting pull requests and issues for bugs/feedback!



SYMBOL	PRICE	CHART
E-Mini S&P 500 Mar 13	1,513.50	
Mini Dow Jones Indus.	13,950.00	



The suite contains controls that address common UI requirements in line-of-business (LOB) applications. These include data management (DataForm), scheduling (Calendar), navigation (RadialMenu), data visualization (Chart), and more. Actually, the list itself is rather long. Have a look:

- RadAutoCompleteBox
- RadBulletGraph
- RadBusyIndicator
- RadCalendar
- RadChart
- RadDataBoundListBox
- RadDataForm
- RadDataGrid
- RadDatePicker and RadTimePicker
- RadExpander
- RadGauge
- RadHexView
- RadHubTile
- RadListView
- RadLoopingList
- RadMap
- RadNumericBox
- RadPagination
- RadRadialMenu
- RadRangeSlider
- RadRating
- RadSideDrawer

Not bad, eh?

OPINIONS

PODCAST

Changing Careers to Coding

January 22, 2018

OPINION

5 Things That Actually Scare Developers (and One That Does Not)

October 30, 2017

TELERIK ANNOUNCEMENTS

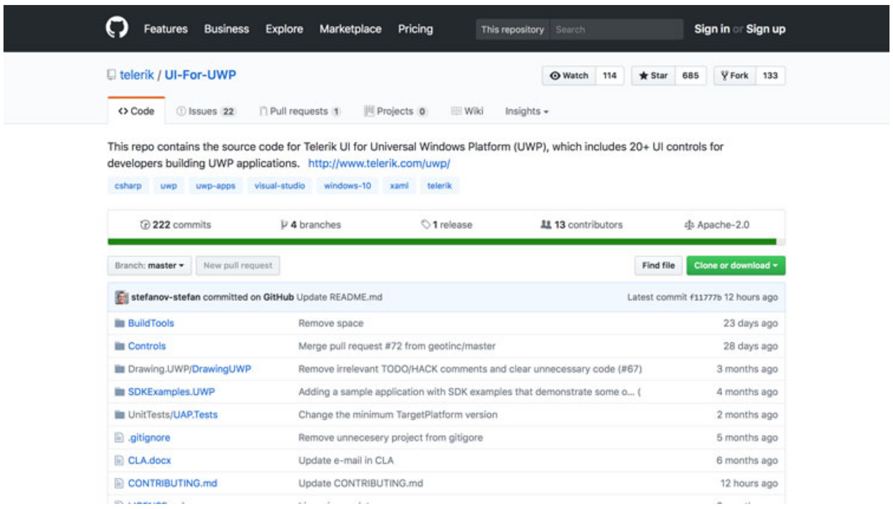
Saying Goodbye to the Telerik Developer Network

Progress Telerik UI for UWP Joins the .NET Foundation

FOLLOW US

In this article, I'll walk you through the source code and show you how to build an application using the **Grid** and **Chart** controls as an example.

Understanding the Repository Structure



telerik / UI-For-UWP

This repo contains the source code for Telerik UI for Universal Windows Platform (UWP), which includes 20+ UI controls for developers building UWP applications. http://www.telerik.com/uwp/

csharp uwp uwp-apps visual-studio windows-10 xaml telerik

222 commits 4 branches 1 release 13 contributors Apache-2.0

stefanov-stefan committed on GitHub Update README.md Latest commit f11777b 12 hours ago

BuildTools	Remove space	23 days ago
Controls	Merge pull request #72 from geotinc/master	28 days ago
Drawing.UWP/DrawingUWP	Remove irrelevant TODO/HACK comments and clear unnecessary code (#67)	3 months ago
SDKExamples.UWP	Adding a sample application with SDK examples that demonstrate some o... (	4 months ago
UnitTests/UAP.Tests	Change the minimum TargetPlatform version	2 months ago
.gitignore	Remove unnecesery project from gitigore	5 months ago
CLA.docx	Update e-mail in CLA	6 months ago
CONTRIBUTING.md	Update CONTRIBUTING.md	12 hours ago

The source code for Telerik UI for UWP is published on GitHub and is structured as follows:

- `BuildTools`: contains scripts and configs for building **Telerik UI for UWP** as a local NuGet package
- `Controls`: contains source files for **Telerik UI for UWP**
- `Drawing.UWP/DrawingUWP`: Direct2D/C++ project used by the map component for rendering shapes
- `SDKExamples.UWP`: contains over 190 examples of the **Telerik UI for UWP** control suite
- `UnitTests/UAP.Tests`: contains unit tests for controls like the **Chart** and **Grid**

The repository has two solution files, `UWPControls.sln` and `UWPMap.sln`. These contain the source files of the control suite and map component. There's also a project file, `SDKExamples.UWP.csproj` that contains the SDK examples. I'd recommend walking through the XAML and C# source files. They contain many examples to help you understand how the controls operate.

Building from Source

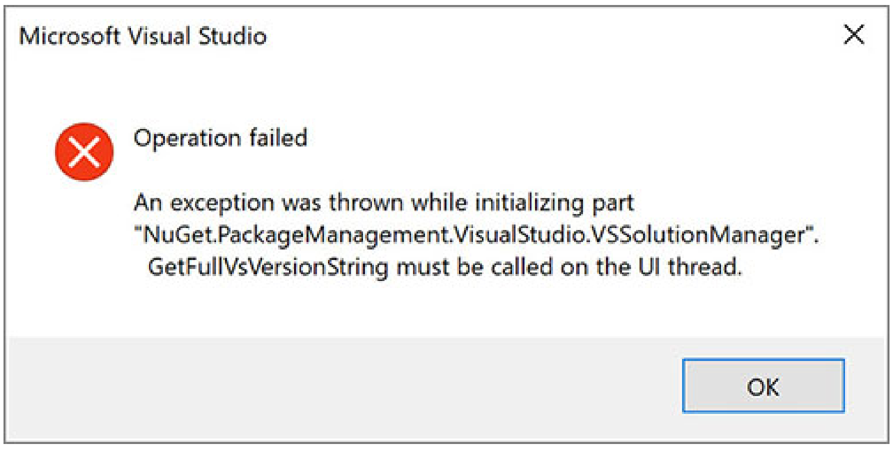
Before you start getting your hands dirty with **Telerik UI**

for UWP, it's important that you have the required software installed to build UWP applications. If you don't, Visual Studio will get angry and take the ball home (so-to-speak). Please note that both **Visual Studio 2015** and **Visual Studio 2017** are supported. That stated, I'd recommend using **Visual Studio 2017** due to the recent improvements that have been added to the IDE. It's always best more fun on the latest and greatest anyway.

The easiest way to build the source of **Telerik UI for UWP** is to open the file, `UWPControls.sln` in Visual Studio and compile the solution from there. Under its default configuration, Visual Studio will restore the NuGet packages that are used by **Telerik UI for UWP** during this process. The same procedure should be followed for building the map component. This is located in the solution, `UWPMap.sln`. In both cases, assemblies (along with PDBs and compiled resource files) will be created in a folder labelled, `Binaries` after the build process completes.

A NOTE ON BUILDING THE SOURCE CODE

You may encounter an error when attempting to build the source with Visual Studio:


Microsoft Visual Studio

Operation failed

An exception was thrown while initializing part "NuGet.PackageManagement.VisualStudio.VSSolutionManager". GetFullVsVersionString must be called on the UI thread.

The following workaround will resolve this behaviour:

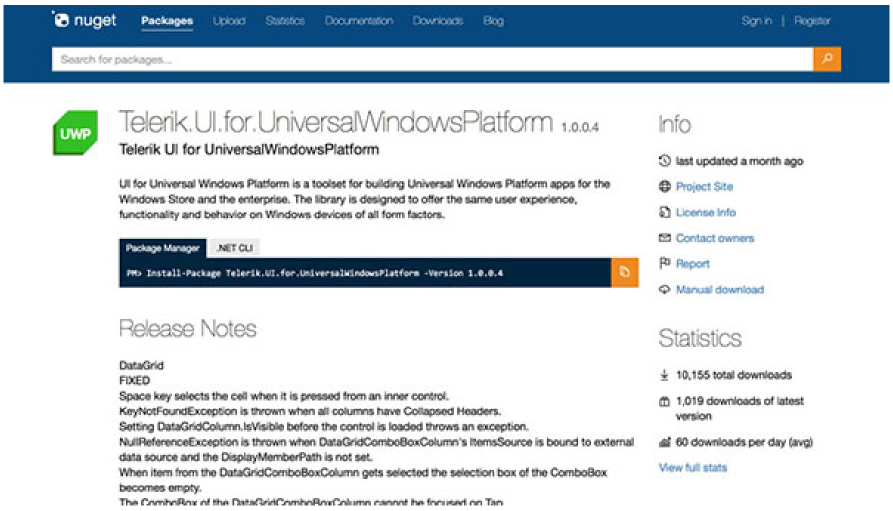
1. Before opening a solution/project, open the **NuGet Package Manager Settings**
2. Wait for the settings dialog to appear and click OK
3. Open your solution/project and build it (NuGet will

work until the next time Visual Studio is restarted)

Credit to **Hrvoje Matić** for discovering this workaround, which was posted to the NuGet issues list back in September 2016.

Using the Public NuGet Package

Telerik UI for UWP is available as a package on NuGet and can be incorporated into a new or existing product through Visual Studio:



Telerik.UI.for.UniversalWindowsPlatform 1.0.0.4

Telerik UI for UniversalWindowsPlatform

UI for Universal Windows Platform is a toolset for building Universal Windows Platform apps for the Windows Store and the enterprise. The library is designed to offer the same user experience, functionality and behavior on Windows devices of all form factors.

Package Manager | .NET CLI

```
PM> Install-Package Telerik.UI.for.UniversalWindowsPlatform -Version 1.0.0.4
```

Release Notes

DataGrid
FIXED
Space key selects the cell when it is pressed from an inner control.
KeyNotFoundException is thrown when all columns have Collapsed Headers.
Setting DataGridColumn.IsVisible before the control is loaded throws an exception.
NullReferenceException is thrown when DataGridComboBoxColumn's ItemsSource is bound to external data source and the DisplayMemberPath is not set.
When item from the DataGridComboBoxColumn gets selected the selection box of the ComboBox becomes empty.
The ComboBox of the DataGridComboBoxColumn cannot be focused on Tap

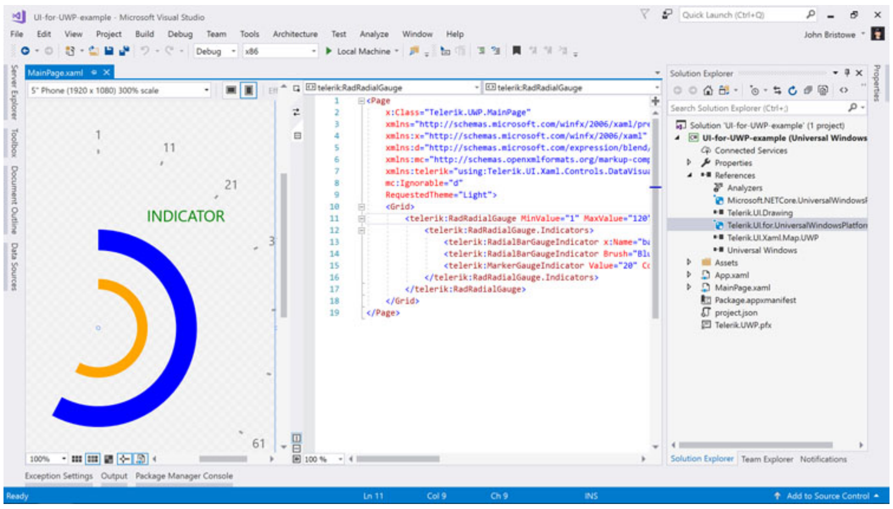
Info

last updated a month ago
Project Site
License Info
Contact owners
Report
Manual download

Statistics

10,155 total downloads
1,019 downloads of latest version
60 downloads per day (avg)
View full stats

You can add this package to a project through the **Package Manager Console** or the **Solution Explorer**:



```
<Page
    x:Class="Telerik.UWP.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/pre
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend,
    xmlns:mc="http://schemas.openxmlformats.org/markup-comp
    xmlns:telerik="using:Telerik.UI.Xaml.Controls.DataVisu
    mc:Ignorable="d"
    RequestedTheme="Light">
    <Grid>
        <telerik:RadRadialGauge MinValue="1" MaxValue="120"
            <telerik:RadRadialGauge.Indicators>
                <telerik:RadialBarGaugeIndicator x:Name="ba
                <telerik:RadialBarGaugeIndicator Brush="Blu
                <telerik:MarkerGaugeIndicator Value="20" Co
            </telerik:RadRadialGauge.Indicators>
        </telerik:RadRadialGauge>
    </Grid>
</Page>
```

Once added, controls can be added and qualified in XAML pages through prefixes and namespaces. Here's an example using the **RadialGauge** control:

```
<Page xmlns:telerik="using:Telerik.UI.Xaml.Controls.Data\
    ...
    <telerik:RadRadialGauge>
      ...
    </telerik:RadRadialGauge>
  </Page>
```

Adding Telerik UI for UWP to Visual Studio Toolbox

At the time of this writing, we don't have an installer to place the **Telerik UI for UWP** controls into the Visual Studio Toolbox. However, you can do this yourself by following these steps:

1. Add the `Telerik.UI.for.UniversalWindowsPlatform` NuGet package to your application
2. Open any XAML page and navigate to the **Toolbox** window of Visual Studio (Ctrl+Alt+X)
3. Right-click in a blank area and select, "Add Tab"
4. Specify the tab to something meaningful (i.e. "Telerik UI for UWP")
5. Right-click in this tab and select, "Choose Items…"
6. Click the Browse button in the **Choose Toolbox Items** dialog and navigate to the following folder on your machine: `C:\Users\%USERNAME%\.nuget\packages\telerik.ui.for.universalwindowsplatform`

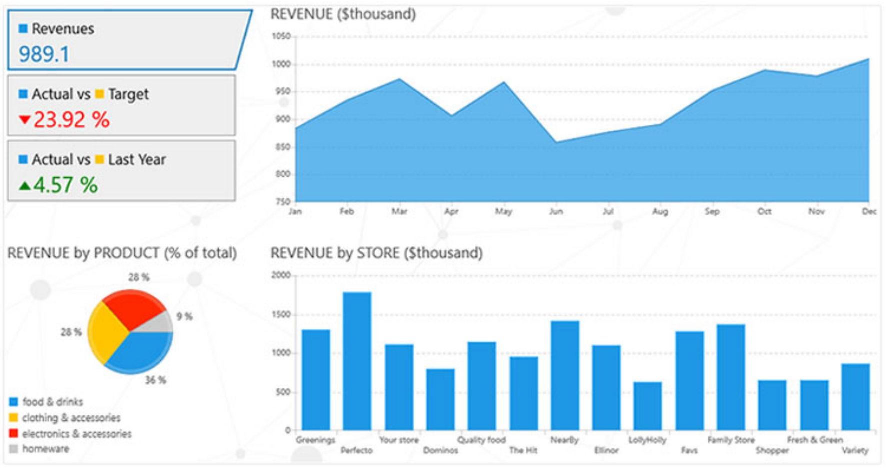
Choose the folder name matching the version you have installed (i.e. `1.0.0.4`), navigate to the `lib\uap10.0` sub-folder and select the following files:

- `Telerik.UI.Xaml.Chart.UWP.dll`
- `Telerik.UI.Xaml.Grid.UWP.dll`
- `Telerik.UI.Xaml.Input.UWP.dll`
- `Telerik.UI.Xaml.Primitives.UWP.dll`
- `Telerik.UI.Xaml.Map.UWP.dll` (located in the `x86`/`x64` sub-folder)

Once added, these controls will be able in the **Toolbox** window of Visual Studio. This will allow you to drag & drag them onto the visual design surface of the XAML pages in your UWP application.

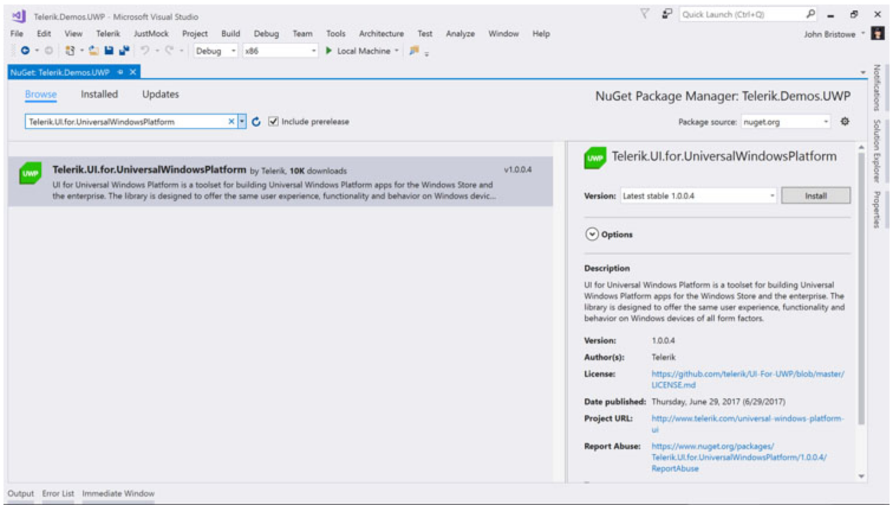
Let's Build an Application

Now, the fun begins: building an application that uses the **Chart** and **Grid** controls. These controls are well understood by developers so this is a good place to start.



The **Chart** control is versatile charting component that you can use to visualize data in many different ways. Here, I'm going to build an application that uses it to represent a set of arbitrary data.

I'll start by creating a new UWP application in Visual Studio and add the **Telerik UI for UWP** NuGet package to the solution:



NuGet Package Manager: Telerik.Demos.UWP

Telerik.UI.for.UniversalWindowsPlatform ☑ Include prerelease

Package source: nuget.org

Telerik.UI.for.UniversalWindowsPlatform by Telerik, **10K** downloads v1.0.0.4
UI for Universal Windows Platform is a toolset for building Universal Windows Platform apps for the Windows Store and the enterprise. The library is designed to offer the same user experience, functionality and behavior on Windows devic...

Telerik.UI.for.UniversalWindowsPlatform

Version: Latest stable 1.0.0.4

Description

UI for Universal Windows Platform is a toolset for building Universal Windows Platform apps for the Windows Store and the enterprise. The library is designed to offer the same user experience, functionality and behavior on Windows devices of all form factors.

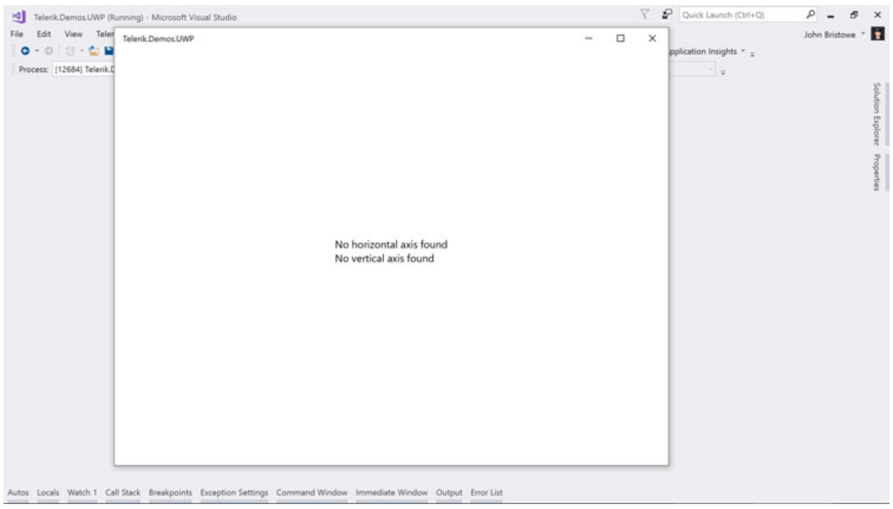
Version:	1.0.0.4
Author(s):	Telerik
License:	https://github.com/telerik/UI-For-UWP/blob/master/LICENSE.md
Date published:	Thursday, June 29, 2017 (6/29/2017)
Project URL:	http://www.telerik.com/universal-windows-platform-ui
Report Abuse:	https://www.nuget.org/packages/Telerik.UI.for.UniversalWindowsPlatform/1.0.0.4/ReportAbuse

From here, I'll open up MainPage.xaml and add a **RadCartesianChart** element to the page:

```
<Page
  x:Class="Telerik.Demos.UWP.MainPage"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/pre
```

```
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/
    xmlns:mc="http://schemas.openxmlformats.org/markup-comp
    xmlns:telerik="using:Telerik.UI.Xaml.Controls.Chart"
    mc:Ignorable="d">
      <Grid Background="{ThemeResource ApplicationPageBack
        <telerik:RadCartesianChart PaletteName="DefaultLigh
        </telerik:RadCartesianChart>
      </Grid>
  </Page>
```

Running this application will display the chart with the following messages:



```
No horizontal axis found
No vertical axis found
```

These messages are displayed by the **Chart** control if you don't define its horizontal and vertical axis. They are required (along with a series definition) to render the data that's associated with the chart at runtime.

The next task is to bind some underlying data and define the horizontal and vertical axis that will be used to display it. I'll define a model that will generate a collection of random values for Australian city names to which I can bind the chart:

```
  public class City
  {
    public string Name { get; set; }
    public double Value { get; set; }
  }

  public class CityManager
  {
    private static Random random = new Random();
    private static string[] cityNames = new string[] { "Mel
    public static List<City> GetCities()
    {
      List<City> cities = new List<City>();
      for (int i = 0; i < cityNames.Length; i++)
      {
        cities.Add(new City() { Name = cityNames[i], Value
      }
      return cities;
    }
  }
```

Next, I'll add to the chart declaration to include the axis and a line series:

```
  <Page
    x:Class="Telerik.Demos.UWP.MainPage"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/p
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/
    xmlns:mc="http://schemas.openxmlformats.org/markup-comp
    xmlns:telerik="using:Telerik.UI.Xaml.Controls.Chart"
    mc:Ignorable="d">
      <Grid Background="{ThemeResource ApplicationPageBackg
        <telerik:RadCartesianChart PaletteName="DefaultLigh
          <telerik:RadCartesianChart.HorizontalAxis>
            <telerik:CategoricalAxis />
          </telerik:RadCartesianChart.HorizontalAxis>
          <telerik:RadCartesianChart.VerticalAxis>
            <telerik:LinearAxis />
          </telerik:RadCartesianChart.VerticalAxis>
          <telerik:LineSeries ItemsSource="{Binding}">
            <telerik:LineSeries.ValueBinding>
              <telerik:PropertyNameDataPointBinding Propert
            </telerik:LineSeries.ValueBinding>
            <telerik:LineSeries.CategoryBinding>
              <telerik:PropertyNameDataPointBinding Propert
            </telerik:LineSeries.CategoryBinding>
          </telerik:LineSeries>
        </telerik:RadCartesianChart>
      </Grid>
  </Page>
```

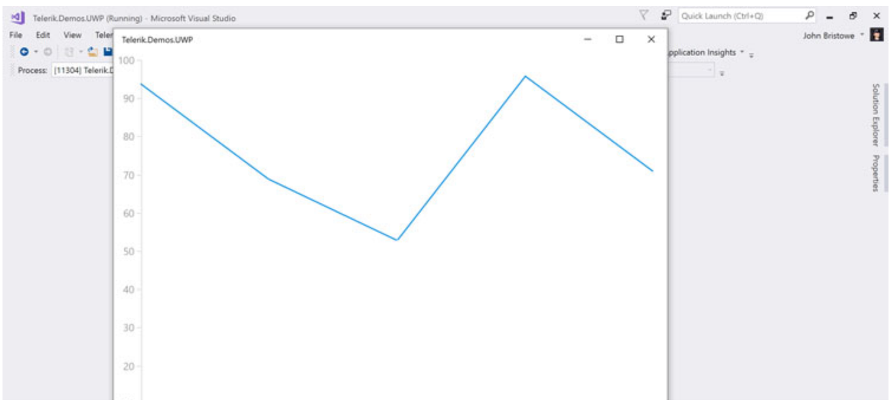
In the XAML (above), I've defined a **CategoricalAxis** for the horizontal axis. This displays a range of categories whose values are displayed in the order defined by the underlying collection that's bound to the chart. The vertical axis uses a **LinearAxis** which represents a sequential list of numerical values. Other axis types for date/time values and logarithmic sequences are also supported. I've also indicated that I've like to generate a line chart based on the **LineSeries** that's declared. Notice that I've bound the value and category values to properties found on the **City** class I defined earlier. Other options to binding data are available, such as specifying the data points in the XAML directly:

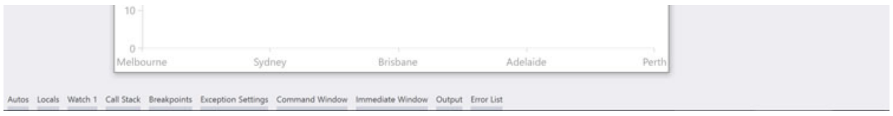
```
<telerik:RadCartesianChart PaletteName="DefaultLight">
  <telerik:RadCartesianChart.Series>
    <telerik:LineSeries>
      <telerik:LineSeries.DataPoints>
        <telerik:CategoricalDataPoint Category="Melbourne
        <telerik:CategoricalDataPoint Category="Sydney" V
        <telerik:CategoricalDataPoint Category="Brisbane"
        <telerik:CategoricalDataPoint Category="Adelaide"
        <telerik:CategoricalDataPoint Category="Perth" Va
      </telerik:LineSeries.DataPoints>
    </telerik:LineSeries>
  </telerik:RadCartesianChart.Series>
  <!-- markup removed for brevity -->
</telerik:RadCartesianChart>
```

The final step is to bind the chart to the data that's generated by the **CityManager** class in the constructor of the page:

```
public sealed partial class MainPage : Page
{
  public MainPage()
  {
    this.InitializeComponent();
    this.DataContext = CityManager.GetCities();
  }
}
```

his application will display the chart of the data domly generated:

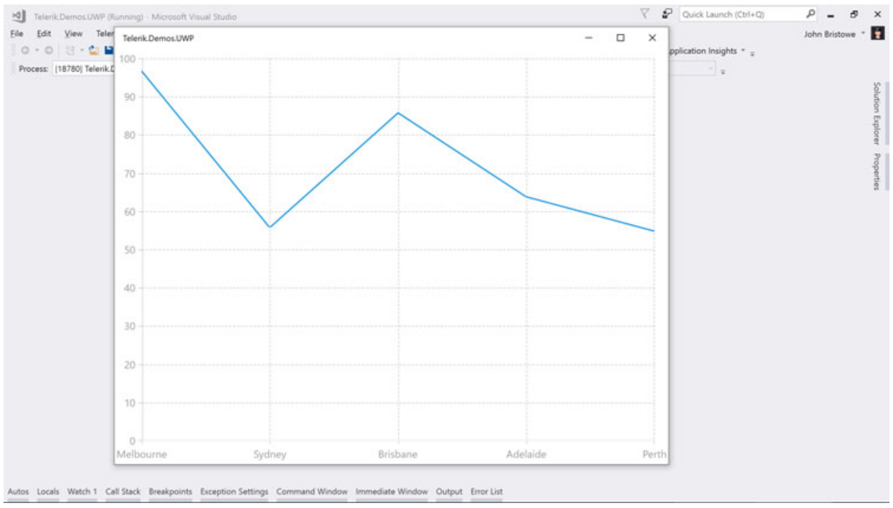



10
0
Melbourne
Sydney
Brisbane
Adelaide
Perth
Autos Locals Watch 1 Call Stack Breakpoints Exception Settings Command Window Immediate Window Output Error List

It's important to note that the **Chart** control is highly customizable. There are a number of properties that I can target that will modify the chart that's generated. For example, I can add the following XAML to my chart declaration to draw lines for the major axis points:

```
<telerik:RadCartesianChart PaletteName="DefaultLight">
  <telerik:RadCartesianChart.Grid>
    <telerik:CartesianChartGrid MajorLinesVisibility="XY'
  </telerik:RadCartesianChart.Grid>
  <!-- markup removed for brevity -->
</telerik:RadCartesianChart>
```

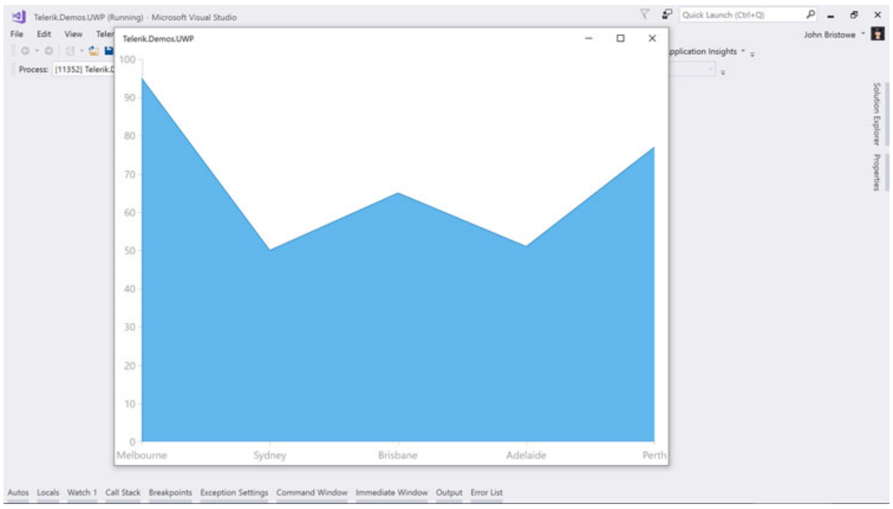
This change will display the chart with the X and Y-axis major lines drawn:


Telerik.Demos.UWP (Running) - Microsoft Visual Studio
Quick Launch (Ctrl+Q)
File Edit View
Process: [18780] Telerik.D
Telerik.Demos.UWP
pplication Insights
John Bristowe
Solution Explorer
Properties
100
90
80
70
60
50
40
30
20
10
0
Melbourne
Sydney
Brisbane
Adelaide
Perth
Autos Locals Watch 1 Call Stack Breakpoints Exception Settings Command Window Immediate Window Output Error List

From here, you can modify the XAML used to generate a chart to display an area chart:

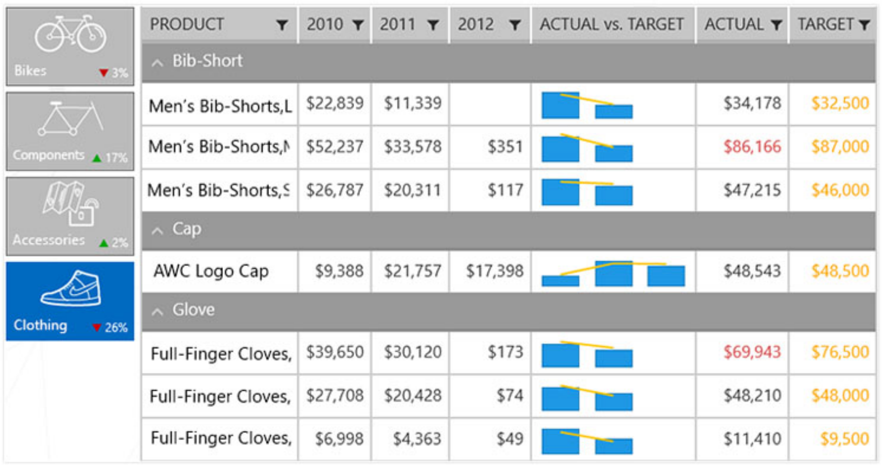
```
<telerik:RadCartesianChart PaletteName="DefaultLight">
  <telerik:AreaSeries ItemsSource="{Binding}">
    <telerik:AreaSeries.ValueBinding>
      <telerik:PropertyNameDataPointBinding PropertyName=
    </telerik:AreaSeries.ValueBinding>
    <telerik:AreaSeries.CategoryBinding>
      <telerik:PropertyNameDataPointBinding PropertyName=
    </telerik:AreaSeries.CategoryBinding>
  </telerik:AreaSeries>
  <!-- markup removed for brevity -->
</telerik:RadCartesianChart>
```

Running the application with this change will generate an area chart a different set of random data:



As you delve into the specifics of the **Chart** control from **Telerik UI for UWP**, you'll discover that it has many built-in features. You can read more about these features on the for the **Chart** control in our documentation.

Now, let's see an example of the **Grid** in **Telerik UI for UWP** using the same underlying data.



PRODUCT	2010	2011	2012	ACTUAL vs. TARGET	ACTUAL	TARGET
Bib-Short						
Men's Bib-Shorts,L	$22,839	$11,339			$34,178	$32,500
Men's Bib-Shorts,M	$52,237	$33,578	$351		$86,166	$87,000
Men's Bib-Shorts,S	$26,787	$20,311	$117		$47,215	$46,000
Cap						
AWC Logo Cap	$9,388	$21,757	$17,398		$48,543	$48,500
Glove						
Full-Finger Cloves,	$39,650	$30,120	$173		$69,943	$76,500
Full-Finger Cloves,	$27,708	$20,428	$74		$48,210	$48,000
Full-Finger Cloves,	$6,998	$4,363	$49		$11,410	$9,500

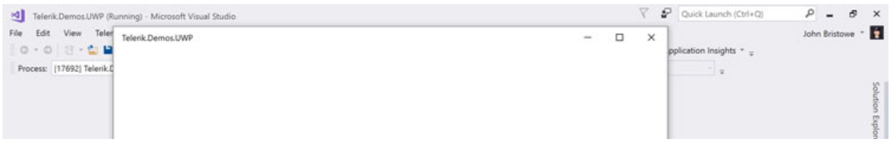
The **Grid** control enables you to display and manipulate vast amounts of data in a control that's fast, fluid, and responsive. It's also highly customizable for operations like filtering, paging, sorting, and grouping. Let's see how to get a **Grid** control up and running in my existing UWP application.

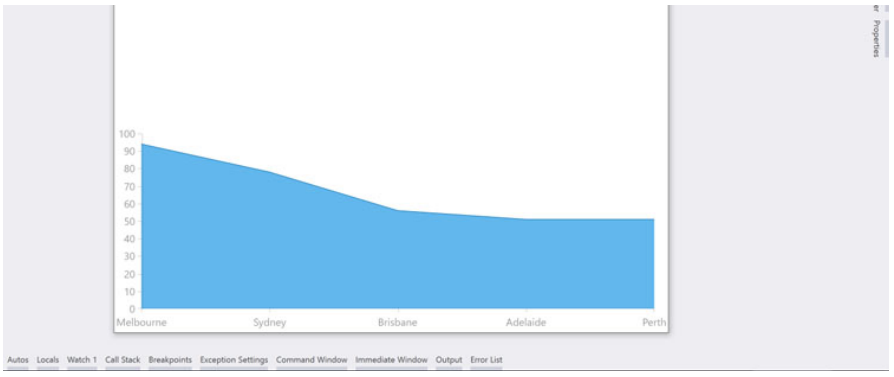
I'll start by making a few modifications to the XAML I defined earlier:

```
<Page
  x:Class="Telerik.Demos.UWP.MainPage"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/pre
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  xmlns:d="http://schemas.microsoft.com/expression/blend/
  xmlns:mc="http://schemas.openxmlformats.org/markup-comp
  xmlns:telerikChart="using:Telerik.UI.Xaml.Controls.Char
  mc:Ignorable="d">
    <Grid Background="{ThemeResource ApplicationPageBackg
      <!-- added: column definitions for layout -->
      <Grid.RowDefinitions>
        <RowDefinition />
        <RowDefinition />
      </Grid.RowDefinitions>
      <!-- I'll add the grid here -->
      <!-- added: specified column location -->
      <telerikChart:RadCartesianChart Grid.Row="1" Palett
        <telerikChart:RadCartesianChart.HorizontalAxis>
          <telerikChart:CategoricalAxis />
        </telerikChart:RadCartesianChart.HorizontalAxis>
        <telerikChart:RadCartesianChart.VerticalAxis>
          <telerikChart:LinearAxis />
        </telerikChart:RadCartesianChart.VerticalAxis>
        <telerikChart:AreaSeries ItemsSource="{Binding}">
          <telerikChart:AreaSeries.ValueBinding>
            <telerikChart:PropertyNameDataPointBinding Pr
          </telerikChart:AreaSeries.ValueBinding>
          <telerikChart:AreaSeries.CategoryBinding>
            <telerikChart:PropertyNameDataPointBinding Pr
          </telerikChart:AreaSeries.CategoryBinding>
        </telerikChart:AreaSeries>
      </telerikChart:RadCartesianChart>
    </Grid>
</Page>
```

In the XAML (above), I've made a couple of changes. First, I've changed the namespace prefix, `telerik` to `telerikChart`. I'll soon add another namespace for the **Grid** control and I want a better naming convention to identify my controls. And second, I've specified the layout of the grid used to contain my controls to include a couple of row definitions. Essentially, I want the **Grid** control to sit above the existing **Chart** control on my page.

Running this code displays our application with a modified layout:





Now let's add the **Grid** control.

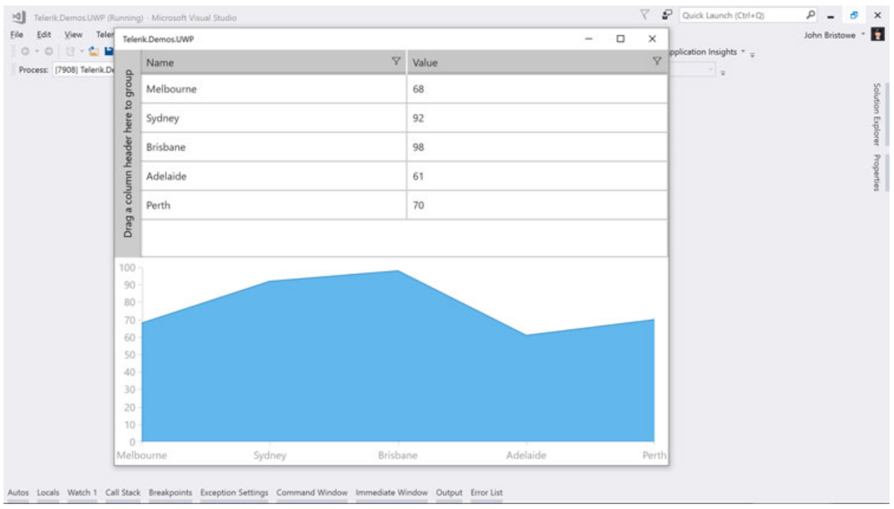
As it turns out, this is a pretty simple task:

```
<telerikGrid:RadDataGrid Grid.Row="0" ItemsSource="{Bindi
</telerikGrid:RadDataGrid>
```

This is all that's required to add to our existing markup in order to have a **Grid** control added to our existing application. Here's how the page looks with this change:

```
<Page
  x:Class="Telerik.Demos.UWP.MainPage"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/pre
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  xmlns:d="http://schemas.microsoft.com/expression/blend/
  xmlns:mc="http://schemas.openxmlformats.org/markup-comp
  xmlns:telerikChart="using:Telerik.UI.Xaml.Controls.Char
  xmlns:telerikGrid="using:Telerik.UI.Xaml.Controls.Grid'
  mc:Ignorable="d">
    <Grid Background="{ThemeResource ApplicationPageBackg
      <Grid.RowDefinitions>
        <RowDefinition />
        <RowDefinition />
      </Grid.RowDefinitions>
      <telerikGrid:RadDataGrid Grid.Row="0" ItemsSource='
      </telerikGrid:RadDataGrid>
      <telerikChart:RadCartesianChart Grid.Row="1" Palett
        <telerikChart:RadCartesianChart.HorizontalAxis>
          <telerikChart:CategoricalAxis />
        </telerikChart:RadCartesianChart.HorizontalAxis>
        <telerikChart:RadCartesianChart.VerticalAxis>
          <telerikChart:LinearAxis />
        </telerikChart:RadCartesianChart.VerticalAxis>
        <telerikChart:AreaSeries ItemsSource="{Binding}">
          <telerikChart:AreaSeries.ValueBinding>
            <telerikChart:PropertyNameDataPointBinding Pr
          </telerikChart:AreaSeries.ValueBinding>
          <telerikChart:AreaSeries.CategoryBinding>
            <telerikChart:PropertyNameDataPointBinding Pr
          </telerikChart:AreaSeries.CategoryBinding>
        </telerikChart:AreaSeries>
        </telerikChart:RadCartesianChart>
    </Grid>
</Page>
```

You can start to appreciate just how powerful these controls are when you run the application:



Name	Value
Melbourne	68
Sydney	92
Brisbane	98
Adelaide	61
Perth	70

Telerik.Demos.UWP
Drag a column header here to group
100
90
80
70
60
50
40
30
20
10
0
Melbourne
Sydney
Brisbane
Adelaide
Perth
Autos Locals Watch 1 Call Stack Breakpoints Exception Settings Command Window Immediate Window Output Error List

By default, the **Grid** auto-generates the columns based on the object that it's bound to. This includes built-in filtering, which you can use to drill down into the data. Furthermore, the **Grid** control has grouping enabled by default. This allows me to combine rows that have equal column values. Let's modify the code that's used to generate this data to see how this works:

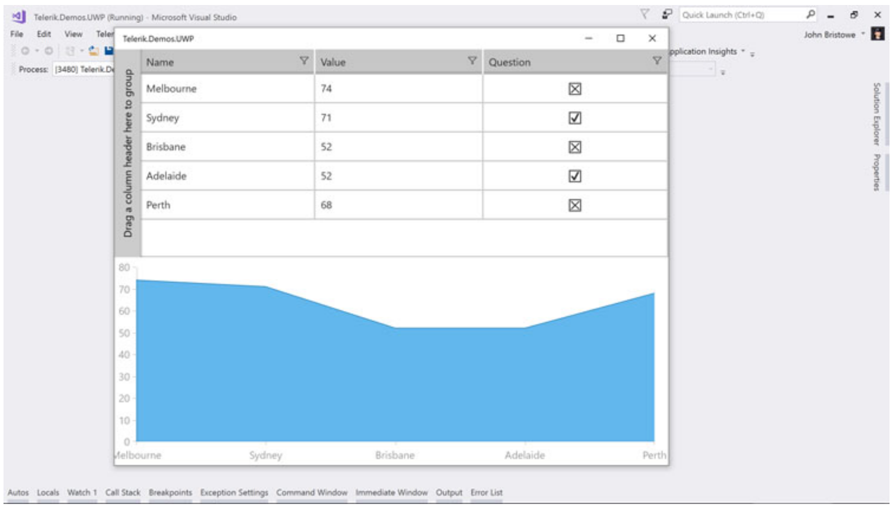
```
public class City
{
  public string Name { get; set; }
  public double Value { get; set; }
  public bool Question { get; set; }
}

public class CityManager
{
  private static Random random = new Random();
  private static string[] cityNames = new string[] { "Me

  public static List<City> GetCities()
  {
    List<City> cities = new List<City>();
    for (int i = 0; i < cityNames.Length; i++)
    {
      cities.Add(new City()
      {
        Name = cityNames[i],
        Value = random.Next(50, 100),
        Question = random.Next(0, 2) == 0
```

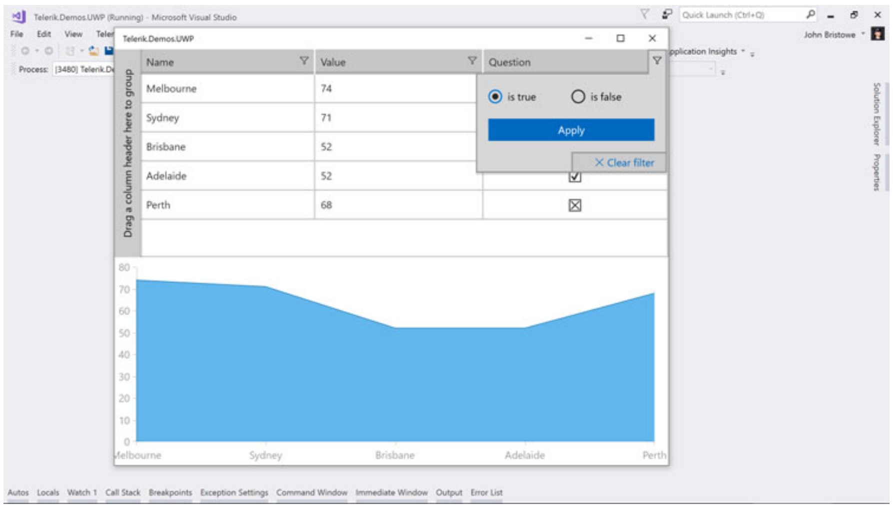
```
            });
        }
        return cities;
    }
}
```

Here, I've added a Boolean property to the **City** class that I've given an arbitrary name. Its value will be randomly generated to be either true or false. When we run our application, notice that this new property is automatically displayed by the **Grid** control:


Telerik.Demos.UWP (Running) - Microsoft Visual Studio
Telerik.Demos.UWP
Drag a column header here to group
Melbourne
Sydney
Brisbane
Adelaide
Perth
Autos Locals Watch 1 Call Stack Breakpoints Exception Settings Command Window Immediate Window Output Error List

Name	Value	Question
Melbourne	74	☒
Sydney	71	☑
Brisbane	52	☒
Adelaide	52	☑
Perth	68	☒

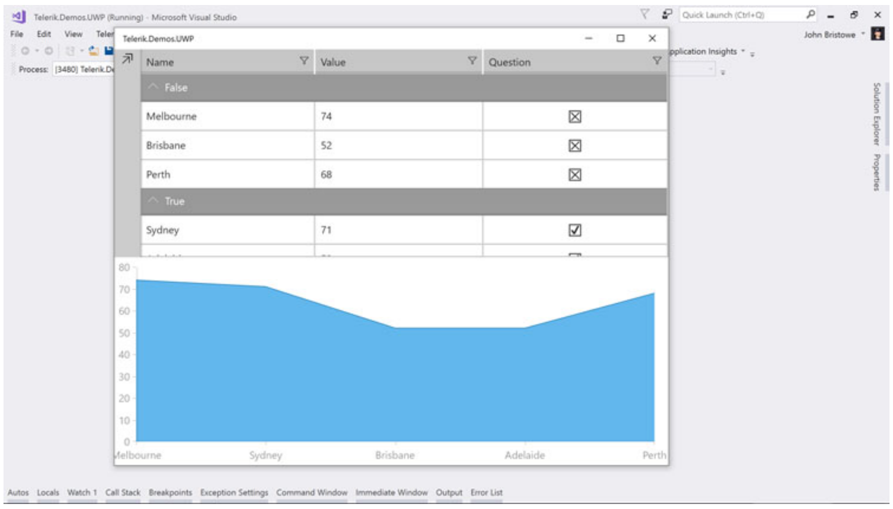
The **Grid** control knows that this new column's type is a Boolean so it also uses a checkbox to display its value. This is also reflected in the filter drop-down menu in the column header:


Telerik.Demos.UWP (Running) - Microsoft Visual Studio
Telerik.Demos.UWP
Drag a column header here to group
[✓ is true] is false
Apply
Clear filter
Melbourne
Sydney
Brisbane
Adelaide
Perth
Autos Locals Watch 1 Call Stack Breakpoints Exception Settings Command Window Immediate Window Output Error List

Name	Value	Question
Melbourne	74	
Sydney	71	
Brisbane	52	
Adelaide	52	☑
Perth	68	☒

The **Grid** control has support built-in for primitives types (i.e. Boolean) for operations like filtering, sorting, and grouping. When binding to complex types, you can control how values are displayed and navigated through these operations as well. If left undefined, the **Grid** control will display the type name for complex types by default.

With grouping enabled by default, we can drag and drop a column header to the grouping area and have the rows grouped by the new property I've added:



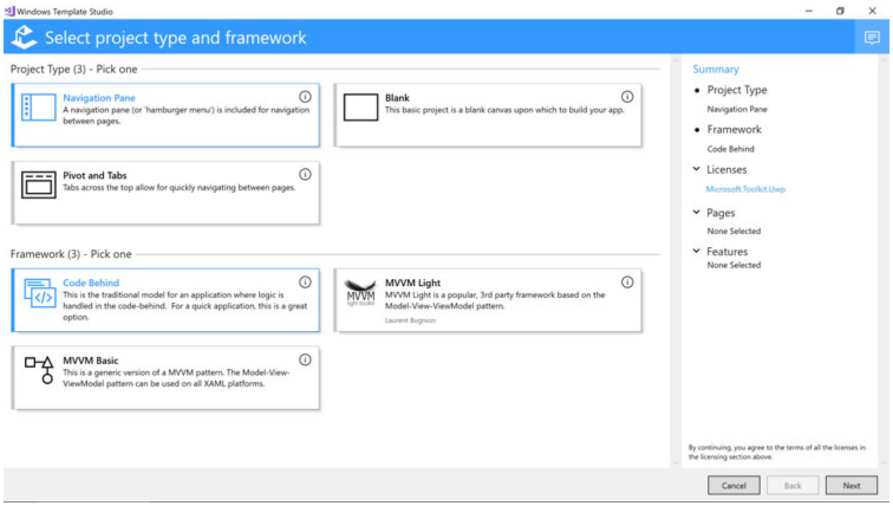
Name	Value	Question
False		
Melbourne	74	☒
Brisbane	52	☒
Perth	68	☒
True		
Sydney	71	☑

Telerik.Demos.UWP (Running) - Microsoft Visual Studio
Telerik.Demos.UWP
Quick Launch (Ctrl+Q)
John Bristowe
Application Insights
Solution Explorer
Properties
80
70
60
50
40
30
20
10
0
Melbourne
Sydney
Brisbane
Adelaide
Perth
Autos Locals Watch 1 Call Stack Breakpoints Exception Settings Command Window Immediate Window Output Error List

Even when grouping is enabled, operations such as paging, filtering, and sorting remain intact.

Using Telerik UI for UWP in Windows Template Studio

As you've seen, incorporating Telerik UI for UWP into a UWP project is a manual process. Developers looking for a guided approach towards building applications with **Telerik UI for UWP** should check out Windows Template Studio:



Windows Template Studio

Select project type and framework

Project Type (3) - Pick one

Navigation Pane — A navigation pane (or 'hamburger menu') is included for navigation between pages.

Blank — This basic project is a blank canvas upon which to build your app.

Pivot and Tabs — Tabs across the top allow for quickly navigating between pages.

Framework (3) - Pick one

Code Behind — This is the traditional model for an application where logic is handled in the code-behind. For a quick application, this is a great option.

MVVM Light — MVVM Light is a popular, 3rd party framework based on the Model-View-ViewModel pattern. Laurent Bugnion

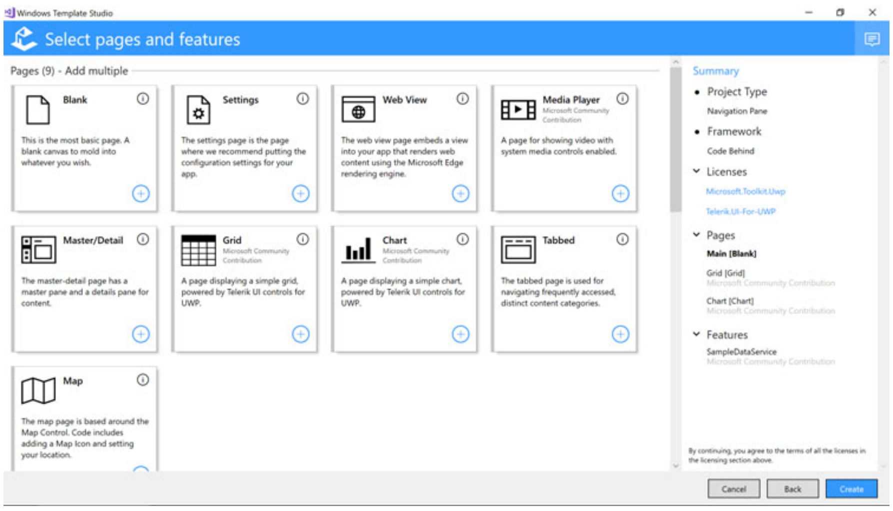
MVVM Basic — This is a generic version of a MVVM pattern. The Model-View-ViewModel pattern can be used on all XAML platforms.

Summary
- Project Type: Navigation Pane
- Framework: Code Behind
- Licenses: Microsoft.Toolkit.Uwp
- Pages: None Selected
- Features: None Selected

By continuing, you agree to the terms of all the licenses in the licensing section above.

This extension for Visual Studio will generate a UWP application through templates. The goal is to get you up and running quickly with a project structure and source files that can be modified afterward.

Recently, Windows Template Studio added templates for the **Chart** and **Grid** controls from Telerik UI for UWP:



Windows Template Studio

Select pages and features

Pages (9) - Add multiple

Blank
This is the most basic page. A blank canvas to mold into whatever you wish.

Settings
The settings page is the page where we recommend putting the configuration settings for your app.

Web View
The web view page embeds a view into your app that renders web content using the Microsoft Edge rendering engine.

Media Player Microsoft Community Contribution
A page for showing video with system media controls enabled.

Master/Detail
The master-detail page has a master pane and a details pane for content.

Grid Microsoft Community Contribution
A page displaying a simple grid, powered by Telerik UI controls for UWP.

Chart Microsoft Community Contribution
A page displaying a simple chart, powered by Telerik UI controls for UWP.

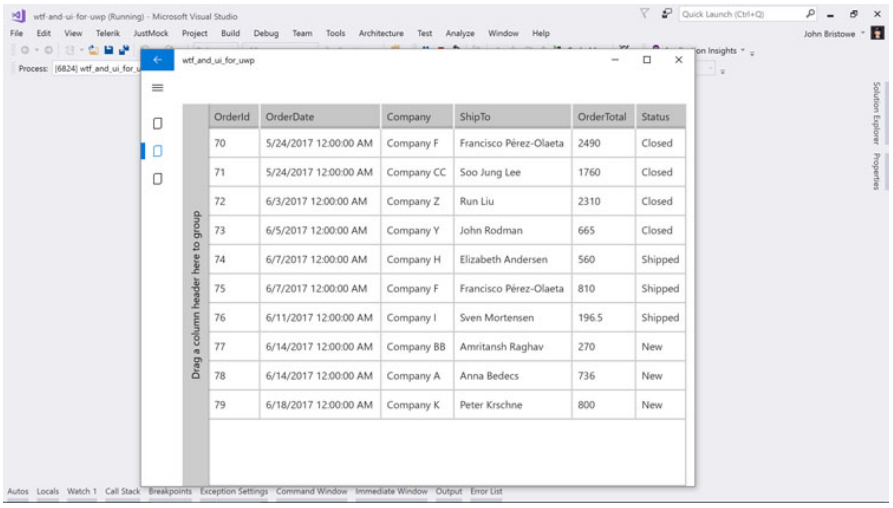
Tabbed
The tabbed page is used for navigating frequently accessed, distinct content categories.

Map
The map page is based around the Map Control. Code includes adding a Map Icon and setting your location.

Summary
- Project Type
  Navigation Pane
- Framework
  Code Behind
- Licenses
  Microsoft.Toolkit.Uwp
  Telerik.UI-For-UWP
- Pages
  Main [Blank]
  Grid [Grid] Microsoft Community Contribution
  Chart [Chart] Microsoft Community Contribution
- Features
  SampleDataService Microsoft Community Contribution

By continuing, you agree to the terms of all the licenses in the licensing section above.

Selecting either of those controls will generate pages will the necessary XAML and code needed to display them:



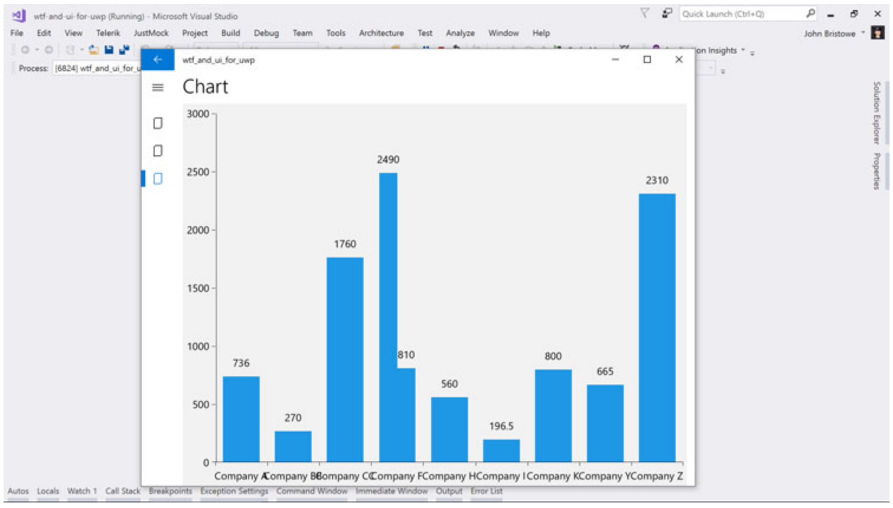
wtf-and-ui-for-uwp (Running) - Microsoft Visual Studio

wtf_and_ui_for_uwp

Drag a column header here to group

OrderId	OrderDate	Company	ShipTo	OrderTotal	Status
70	5/24/2017 12:00:00 AM	Company F	Francisco Pérez-Olaeta	2490	Closed
71	5/24/2017 12:00:00 AM	Company CC	Soo Jung Lee	1760	Closed
72	6/3/2017 12:00:00 AM	Company Z	Run Liu	2310	Closed
73	6/5/2017 12:00:00 AM	Company Y	John Rodman	665	Closed
74	6/7/2017 12:00:00 AM	Company H	Elizabeth Andersen	560	Shipped
75	6/7/2017 12:00:00 AM	Company F	Francisco Pérez-Olaeta	810	Shipped
76	6/11/2017 12:00:00 AM	Company I	Sven Mortensen	196.5	Shipped
77	6/14/2017 12:00:00 AM	Company BB	Amritansh Raghav	270	New
78	6/14/2017 12:00:00 AM	Company A	Anna Bedecs	736	New
79	6/18/2017 12:00:00 AM	Company K	Peter Krschne	800	New

File Edit View Telerik JustMock Project Build Debug Team Tools Architecture Test Analyze Window Help
Quick Launch (Ctrl+Q)
John Bristowe
Solution Explorer
Properties
Autos Locals Watch 1 Call Stack Breakpoints Exception Settings Command Window Immediate Window Output Error List



wtf-and-ui-for-uwp (Running) - Microsoft Visual Studio

wtf_and_ui_for_uwp

Chart

3000
2500
2000
1500
1000
500
0
736
270
1760
2490
810
560
196.5
800
665
2310
Company A
Company BB
Company CC
Company F
Company H
Company I
Company K
Company Y
Company Z
File Edit View Telerik JustMock Project Build Debug Team Tools Architecture Test Analyze Window Help
Quick Launch (Ctrl+Q)
John Bristowe
Solution Explorer
Properties
Autos Locals Watch 1 Call Stack Breakpoints Exception Settings Command Window Immediate Window Output Error List

Wrapping Up

Telerik UI for UWP provides a suite of powerful controls that you can use in your UWP applications. These controls address common UI requirements in line-of-business (LOB) applications. You can find the source code (Apache License v2.0) and the documentation on GitHub.

Getting started is easy. You can build the source code yourself or pull down the bits from NuGet. Once downloaded, controls like the **Chart** and **Grid** can be easily incorporated and bound to data from underlying sources.

If you have feedback, please let us know through the Telerik UI for UWP issues list on GitHub or the Telerik UI for UWP Feedback Portal. In the meantime, I encourage you to check the controls, build a prototype, and/or start incorporating them into your UWP apps today!

← PREV
XAML Standard Demystified

NEXT →
How to Launch Android Emulators and iOS Simulators From the Command Line

John Bristowe

John Bristowe (@JohnBristowe) is a Principal Developer Advocate with Telerik and lives in Australia. Prior to joining Telerik, he was a Senior Developer Evangelist with Microsoft.

COMMENTS

0 Comments **Telerik Developer Network** 1 **Login**

Recommend **Share** **Sort by Best**

Start the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

Be the first to comment.

ALSO ON **TELERIK DEVELOPER NETWORK**

Building Reusable UI Components in ASP.NET Core

3 comments • 8 months ago

Avatar Ed Charbeneau — "The Telerik Tag Helpers handle a plethora of accessibility concerns for

How to Serve Serverless with Examples in Azure

3 comments • 9 months ago

Avatar Raymond Camden — Thanks for the answers!

Getting Started with TensorFlow for Machine

1 comment • 6 months ago

Avatar Shiva Prasad — Thanks for writing such an informative article. :D

Functions as Data: Functional Programming in C#

22 comments • 2 months ago

Avatar Christopher Hayes — I like this article and I do the exact same thing BUT instead of putting the

BROWSE BY TOPIC

.NET
Cloud
Desktop
IoT & Bots
Machine Learning
Mobile
Progress Labs
Testing
Web

BROWSE BY CONTENT TYPE

Opinion
Podcast
Slack Chats
Tutorials
Video

CONTACT

USA: +1 888 365 2779
UK: +44 13 4436 0444
India: +91 124 4300987
Bulgaria: +359 2 8099850
Australia: +61 3 9805 8670
sales@telerik.com