# Simple Dashboard

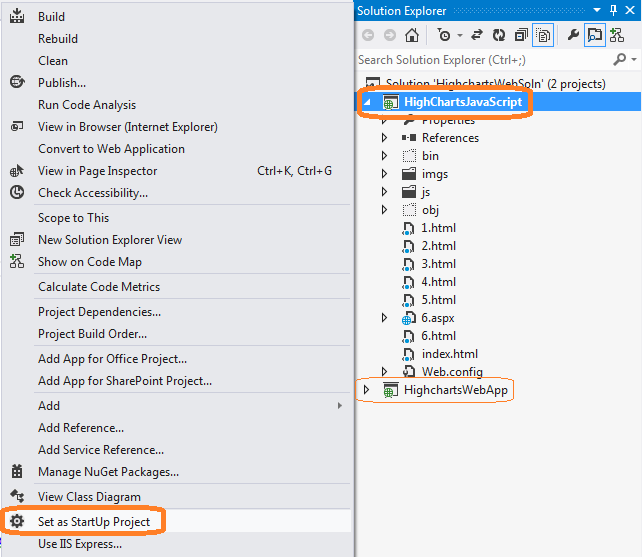
I am a big fan of the [Highcharts charting library](http://www.highcharts.com/) (HCL) and have used it on numerous occasions to create dashboards.

I recently came across the [Highcharts.Net ASP.NET control](http://highcharts.codeplex.com/) (not affiliated to Highcharts) that provides a simple means to encapsulate the Highcharts API.

This article started out as a test project to explore the ASP.NET control as a simpler way to use HCL. While the control and its documentation have some limitations (it does not seem to fully implement HCL and much of the documentation is in Portugese), it does provide a simple option for C# developers who prefer to avoid JavaScript.

The attached solution contains two projects. One project (HighchartsWebApp) demonstrates the use of the ASP.NET control. The second project (HighChartsJavaScript) demonstrates HCL in most of its JavaScript glory.

I have implemented both projects in the form of Dashboards as this is a common ask on StackOverflow and other developer forums. The dashboard template is a combination of JavaScript, CSS and HTML that was designed by the people at [TechGYO](http://techgyo.com/index.php/free-metro-ui-html-template-by-techgyo/). I loved the simplicity of the design and decided to leverage it to create this Highcharts Dashboard.

The object of this exercise is to provide an example that can get you started. This is intended for beginners and has not been stress-tested / live-tested at the time of publication. I might use the feedback in the comments to provide future updates as time permits.

You can activate the JavaScript version by right-clicking the project ‘HighChartsJavaScript’ within Solution Explorer and selecting ‘**Set As StartUp Project**’.

You can alternately run the C# version by right-clicking the ‘HigchartsWebApp‘ within Solution Explorer and selecting ‘**Set As StartUp Project**’. Once you have made your choice, press ‘**F5**’ to activate the demo.

I have put in less effort into the C# version () as I expect future development to concentrate on developing the JavaScript version with the addition of some C# code to retrieve and populate data. If you would like to get more info on using a purely server-side implementation like the ‘HighchartsWebApp’ project, you can check the website for the [Highcharts.Net](http://highcharts.codeplex.com/) open source project (<http://highcharts.codeplex.com/>).

## Dashboard - Initial View



The screenshot above shows an annotated view of the initial dashboard screen. Note, the last (green background) section has been intentionally left empty. Clicking on this section brings up the equivalent of a drill down/popup report (see next screenshot).



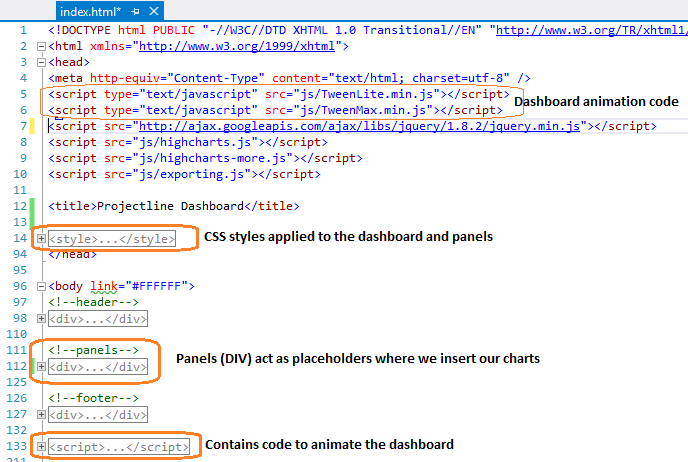
Note, the header changes to reflect a title for the drill down report. The Metro UI themed arrow appears to allow navigation back to the main dashboard.

The above info should be sufficient to guide most of you in getting the project to execute, you could even start modifying it for your individual requirements. For those wanting a deeper dive, let us now look into the code so we can get a clearer vision of its operation.

As mentioned earlier, the attached solution contains two projects, the discussion from this point on focuses on the JavaScript version (‘HighChartsJavaScript’). Our objective is to utilize the Highcharts library to create a dashboard. Additionally, we would like to determine how we can dynamically integrate our data into the charts without having to hard-code it as part of the JavaScript code.

For sake of clarity, I have broken down this review into three parts, part 1 deals with the setup of the dashboard. It examines the HTML, CSS and JavaScript code that enables the look, feel and animation of the dashboard UI. Part 2 will look into the JavaScript code that creates a chart. Part 3 will demonstrate how we can use C# to merge sample application data with the chart code to enable us to integrate our data with the Highcharts library.

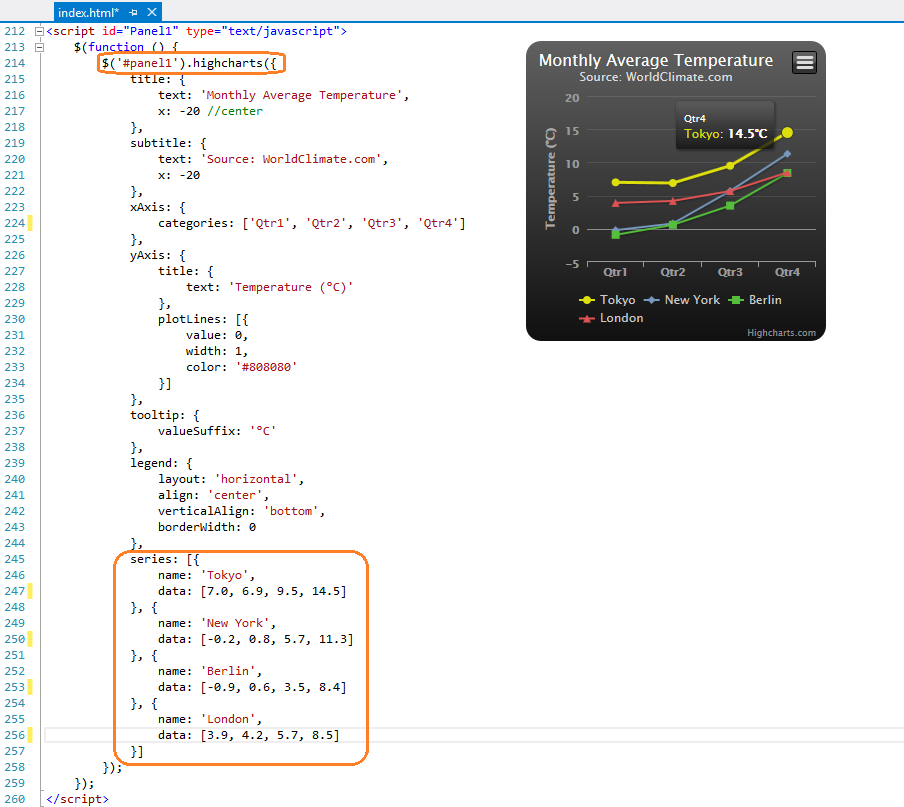
Part 1: Dashboard

The dashboard code as previously mentioned is based on a free template released by the folks at [TechGYO](http://techgyo.com/index.php/free-metro-ui-html-template-by-techgyo/). The majority of the code is contained in the file ‘index.html’, the adjacent screen snippet shows a collapsed view of this file with highlights & labels calling out various functional areas within the code. Note, the code has been collapsed in Visual Studio to provide this bird’s-eye view.

If you examine the code that draws each dashboard panel (lines 112 – 125), you will notice that it consists of a hyperlink enclosing an empty div and not much else. 

This is because all the work of populating the panel is done within JavaScript code blocks that we will discuss in ‘Part 2: Drawing the charts’. The hyperlinks themselves are mostly inert and used only in case of panel 6 (the green panel) to demo drilldown/popup functionality.

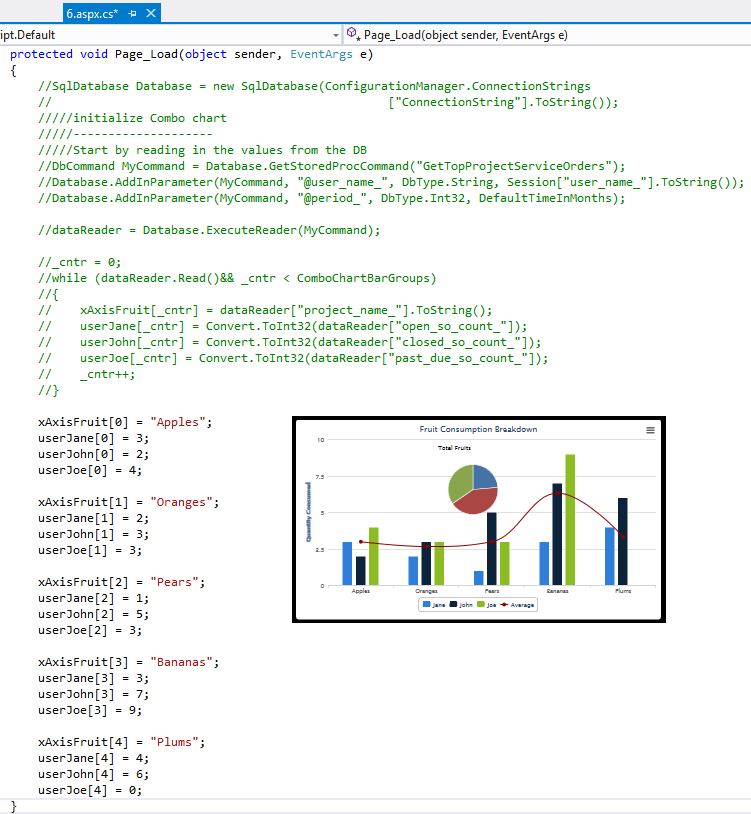
Part 2: Drawing the charts

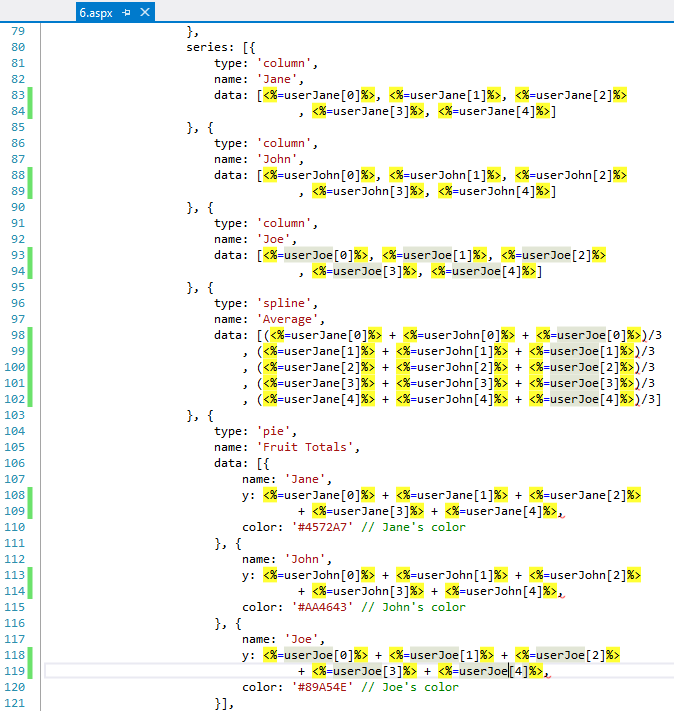
As mentioned in the previous section, each dashboard panel is initially empty and populated by individual JavaScript code blocks. The snippet below shows one such block alongside the chart rendered.

This script defines a JavaScript object. Line 214 ensures that the object (Linechart) is placed inside Panel 1. Lines 245 – 257 define the series data being displayed on the chart. Note, Highchart objects are defined using JSON. If you are unfamiliar with JSON, you should look up one of the many tutorials available on the web. A good starting reference is the one at <http://www.w3schools.com/json/>.

Part 3: Merging application data with chart code

You may have realized that the example until this point is of limited utility as the data is hard-coded into the page. Most use case scenarios involve reading data from a data source and dynamically populating the chart. This section seeks to get you started on that path. In the interest of brevity and focus, I do not plan to cover how data is to be extracted from your data source. I assume you are able to write the code to read your application data. For what it’s worth, I have included some commented code that I used for my application. The item to note in the code below is that we are using some variables (arrays defined at the class-level) to store the values that will be the input for our chart.



You can then bind your application data into JavaScript code using data-binding expressions.

That completes this brief intro to developing a dashboard. You can utilize this framework with any other JavaScript based charting library. You are not limited to charts alone, the content of each dashboard panel can be pretty much anything that is valid within a web page. The simplicity of the TechGYO template will hopefully ensure a short and gentle learning curve. Feel free to throw out ideas on where you would like to take this dashboard, perhaps we can explore it in a future update.

Dependencies:

1. Visual Studio 2012 (<http://www.microsoft.com/visualstudio/eng/downloads>)
2. Highcharts Charting Library (<http://www.highcharts.com/products/highcharts>)
3. Highcharts.Net (<http://highcharts.codeplex.com/>) – not affiliated with Highcharts Charting Library.
4. Metro UI Template by TechGYO.com (<http://techgyo.com/index.php/free-metro-ui-html-template-by-techgyo/>)

Other Dashboard Implementations:

* The Geckoboard Blog (<https://demo.geckoboard.com/dashboard/B6782E562794C2F2/>)
* jSlate (<https://github.com/rasmusbergpalm/jslate/graphs>)
* D3 (<https://github.com/mbostock/d3/wiki/Gallery>)
* Vaadin Dashboard (<https://vaadin.com/demo>)