Configuration Guide - Dashboard Configurator

by Hube Van Loey

This Dashboard Configurator Guide is your go-to resource for crafting tailored, powerful dashboards. Here, we'll explain the process of dashboard creation, offering step-by-step instructions, tips, and best practices. This manual is split up into two parts: The creation of the config.xml file and the .csv-file for loading in the data.

1. The config.xml file

First of all, it is important to configure the config.xml file. In this file you can set up the structure and data visualizations of the entire dashboard, which will then be converted into a real web page by the underlying software.

1.1 Setting up the config.xml file

To set up the whole config.xml, two important tags must be included.

1.1.1 XML tag

The `<?xml?>` tag is used to specify the xml version and the encoding of the file. It is necessary to declare this tag at the top of your config file, so that the software knows this indeed is the right file to handle.

Attributes

- `version=""` | `version="1.0"` | number | required
  The version of xml used. It is advised to use the version given in the example.
- `encoding=""` | `encoding="UTF-8"` | text | required
  The encoding used for the xml file. It is advised to use the encoding given in the example.

1.1.2 Dashboard tag

The `<dashboard>` tag is used to initialize the whole dashboard. Everything between these tags will be visible on the dashboard.

Attributes

The first part of styling the dashboard is done immediatly within the `<dashboard>` tag itself. You're able to declare two colors, primary and secondary, that the software will use to fill in the charts. The reason behind this is that most companies/brands use at most 2 to 3 branding colors. This way, each dashboard can be adjusted so that it is completely in line with the theme of a specific company/brand.

- `primaryColor=""` | `primaryColor="#FFF000"` | hexcode | required
  The primary color for dashboard charts.
- `secondaryColor=""` | `secondaryColor="#FFF000"` | hexcode | required
  The secondary color for dashboard charts.

1.1.3 The starting setup

After defining everything as specified above, the starting setup of the config.xml file will look something like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<dashboard primaryColor="#FF0000" secondaryColor="#0000FF">
  <!-- everything for your dashboard comes here! -->
</dashboard>
```

1.2 Structure Tags

Now that we have declared our dashboard and graph colors, it's time to structure things. To make this possible, we work with a hierarchical structure of tags. There are a couple details you'll have to keep in mind, which will be explained under the 'Best practices' header.

1.2.1 Screen Tag

The `<screen>` tag is used for declaring a whole screen. Let's say you put 3 of these below eachother, you're dashboard will be 3 times the size of the screen you're watching it on.

Attributes

- `color=""` | `color="#FFF000"` | hexcode | optional
  The background color for the screen.
- `title=""` | `title="Title"` | text | optional
  The title text.
- `title-txt-color=""` | `title-txt-color="#FFF000"` | hexcode | optional
  The color for the title text.
- `title-bg-color=""` | `title-bg-color="FFF000"` | hexcode | optional The background color for the title element.

1.2.2 Row Tag

Each `<screen>` tag can be divided into different `row` tags. The software automatically ensures that these rows are equal in height regardless of the number. If only one row is declared (minimum), it will be fit to the size of the parent `screen` tag.

Attributes

- `color=""` | `color="#FFF000"` | hexcode | optional
  The background color for the screen.
- `title=""` | `title="Title"` | text | optional
  The title text.
- `title-txt-color=""` | `title-txt-color="#FFF000"` | hexcode | optional
  The color for the title text.
- `title-bg-color=""` | `title-bg-color="FFF000"` | hexcode | optional
  The background color for the title element.

1.2.3 Section Tag

Each row can then in turn be divided into sections. These sections take on the height of the row they are in. You must set the width of these sections yourself using the `width` attribute.

Attributes

- `width=""` | `width="1/1"` | fraction | required
    - `1/1` (100% screen width)
    - `3/4` (75% screen width)
    - `2/3` (66% screen width)
    - `1/2` (50% screen width)
    - `1/3` (33% screen width)
    - `1/4` (25% screen width)

  The width of the section.

- `color=""` | `color="#FFF000"` | hexcode | optional
  The background color for the screen.

- `title=""` | `title="Title"` | text | optional
  The title text.

- `title-txt-color=""` | `title-txt-color="#FFF000"` | hexcode | optional
  The color for the title text.

- `title-bg-color=""` | `title-bg-color="FFF000"` | hexcode | optional
  The background color for the title element.

It is recommended that you always make sure the sections widths sum up to 100% in total

1.3 Chart Tags

Once the whole layout of the dashboard is set up, all charts can be configured with a `<chart>` tag. Each chart is set up within a `section` - tag and contains different attributes. Some attributes are generally available for all charts, while others are chart-type specific. Below you'll find a list of the attributes you can assign for each chart.

Attributes

- `type=""` | `type="bar"` | chart type | required
  The chart type.
- `title=""` | `title="Title"` | text | optional
  The title for the chart.
- `subtitle=""` | `subtitle="Subtitle"` | text | optional
  The subtitle for the chart.
- `width=""` | `width="50%"` | percentage | optional
  The width of the chart in percentage.
- `height=""` | `height="50%"` | percentage | optional
  The height of the chart in percentage.
- `showLegenda=""` | `showLegenda="false"` (default) | boolean | optional
  - `true` (show legenda)
  - `false` (hide legenda)

  The visability of the legenda. When no value is specified, the default will be used.
- `legendaView=""` | `legendaView="vertical"` (default) | orientation | optional
  - `vertical` (legenda oriented vertically)
  - `horizontal` (legenda oriented horizontally)

  The orientation of the legenda. When no value is specified, the default will be used.

1.3.1 Simple Bar Chart

A basic chart representing data with rectangular bars where the length or height of the bars corresponds to the value they represent. Best used for comparing discrete categories or showing changes over time.

Attributes

- `type="bar"`
  The chart type.
- `targetColumn=""` | `targetColumn="Column name"` | target column name | required
  The column in the csv with the target data (to be visualized).

Item tags

For the simple bar chart, it is possible to manipulate the visualisations even further. Each unique data value should be referred to with an `item` tag.

- `changedName=""` | `changedName="New name"` | new name | optional
  The new name for a specific unique value.

1.3.2 Vertical Multiple Bar Chart

A chart displaying multiple sets of data using grouped bars vertically, allowing for easy comparison between different categories across multiple data sets.

Attributes

- `type="vbar"`
  The chart type.
- `targetColumn=""` | `targetColumn="Column name"` | target column name | required
  The column in the csv with the target data (to be visualized).
- `timeColumn=""` | `timeColumn="Column name"` | time column name | required
  The column in the csv with the time data (used as time reference).

1.3.3 Horizontal Multiple Bar Chart

Similar to the vertical multiple bar chart, but the bars are displayed horizontally, often used when the category labels are long or when space is limited in the vertical direction.

Attributes

- `type="hmbar"`
  The chart type.
- `targetColumn=""` | `targetColumn="Column name"` | target column name | required
  The column in the csv with the target data (to be visualized).
- `timeColumn=""` | `timeColumn="Column name"` | time column name | required
  The column in the csv with the time data (used as time reference).

1.3.4 Stacked Bar Chart

This chart represents multiple data series on top of one another in segments, with each segment representing a proportion of the whole. It's useful for illustrating both total values and how they are divided into different categories.

Attributes

- `type="stackedbar"`
  The chart type.
- `targetColumn=""` | `targetColumn="Column name"` | target column name | required
  The column in the csv with the target data (to be visualized).
- `timeColumn=""` | `timeColumn="Column name"` | time column name | required
  The column in the csv with the time data (used as time reference).

1.3.5 Simple Line Chart

A chart that displays data points connected by straight lines, ideal for showing trends or changes over continuous intervals, such as time.

Attributes

- `type="line"`
  The chart type.
- `targetColumn=""` | `targetColumn="Column name"` | target column name | required
  The column in the csv with the target data (to be visualized).

Item tags

For the simple line chart, it is possible to manipulate the visualisations even further. Each unique data value should be referred to with an `item` tag.

- `changedName=""` | `changedName="New name"` | new name | optional
  The new name for a specific unique value.

1.3.6 Multiple Line Chart

Similar to the simple line chart but displaying multiple lines, often used to compare trends across different data sets.

Attributes

- `type="mline"`
  The chart type.
- `targetColumn=""` | `targetColumn="Column name"` | target column name | required
  The column in the csv with the target data (to be visualized).
- `timeColumn=""` | `timeColumn="Column name"` | time column name | required
  The column in the csv with the time data (used as time reference).

1.3.7 Pie/Donut Chart

A circular chart useful for displaying data with several categories while also showing the overall relationship of each category to the whole.

Attributes

- `type="pie"`
  The chart type.
- `innerCircle=""` | `innerCircle="false"` (default) | boolean | optional
  Indicates wether the chart should have an inner circle (donut chart).
- `targetColumn=""` | `targetColumn="Column name"` | target column name | required
  The column in the csv with the target data (to be visualized).

1.3.8 Radar Chart

A graphical method of displaying multivariate data in the form of a two-dimensional chart with three or more quantitative variables represented on axes starting from the same point, best suited for comparing performance across different categories.

Attributes

- `type="radar"`
  The chart type.
- `targetColumn=""` | `targetColumn="Column name"` | target column name | required
  The column in the csv with the target data (to be visualized).

1.3.9 Single Nightingale Chart

Also known as a polar area diagram or rose chart, it displays data using concentric circles segmented into sectors, with each sector representing a category. Useful for displaying cyclical patterns or proportions within a dataset.

Attributes

- `type="sntgchart"`
  The chart type.
- `targetColumn=""` | `targetColumn="Column name"` | target column name | required
  The column in the csv with the target data (to be visualized).

Item tags

For the single nightingale chart, it is possible to manipulate the visualisations even further. Each unique data value should be referred to with an `item` tag.

- `changedName=""` | `changedName="New name"` | new name | optional
  The new name for a specific unique value.

1.3.10 Multiple Nightingale Chart

Similar to the single nightingale chart but displaying multiple datasets, often used to compare cyclical patterns or proportions across different categories.

Attributes

- `type="mntgchart"`
  The chart type.
- `targetColumn=""` | `targetColumn="Column name"` | target column name | required
  The column in the csv with the target data (to be visualized).
- `timeColumn=""` | `timeColumn="Column name"` | time column name | required
  The column in the csv with the time data (used as time reference).

1.3.11 Scatter Plot

A graph that shows the relationship between two variables by displaying data points on a two-dimensional plane, helpful for identifying correlations or trends between variables.

Attributes

- `type="scatterplot"`
  The chart type.
- `targetColumn=""` | `targetColumn="Column name"` | target column name | required
  The column in the csv with the target data (to be visualized).
- `timeColumn=""` | `timeColumn="Column name"` | time column name | required
  The column in the csv with the time data (used as time reference).

For this chart, there is no distinction between targetColumn and timeColumn behaviour. Since this scatter plot shows the relation between two variables, the data of both columns is used to plot the graph

1.3.12 Scatter Axis

A variation of the scatter plot with axis lines added, making it easier to interpret the relationship between variables in terms of scale or magnitude.

Attributes

- `type="scatteraxis"`
  The chart type.
- `textLeft=""` | `textLeft=""` | text | required
  The text shown on the left of the axis.
- `textRight=""` | `textRight=""` | text | required
  The text shown on the right of the axis.
- `targetColumn=""` | `targetColumn="Column name 1,Column name 2, Column name 3"` | target column name | required
  The columns in the csv with the target data (to be visualized).

In the targetColumn field for the scatter axis, it is possible to pass the names of multiple columns

1.3.13 Gauge Chart

A chart that resembles a speedometer or gauge, typically used to display a single value within a known range, ideal for indicating progress towards a goal or target.

Attributes

- `type="gauge"`
  The chart type.
- `targetColumn=""` | `targetColumn="Column name"` | target column name | required
  The column in the csv with the target data (to be visualized).

1.3.14 Gallery

Although this isn't really a chart, the gallery is defined the same way as the other charts. It can be used to display images via accessiable urls, together with a title and text.

Attributes

- `type="gallery"`
  The chart type.
- `titleColumn=""` | `titleColumn="Column name"` | title column name | optional
  The column in the csv with the title data (to be visualized).
- `textColumn=""` | `textColumn="Column name"` | text column name | optional
  The column in the csv with the text data (to be visualized).
- `imageColumn=""` | `imageColumn="Column name"` | image column name | optional
  The column in the csv with the image data (to be visualized).

1.4 Best Practices

1.4.1 No gaps in hierarchical structure

The current software is set up so that no gaps are tolerated between tags. The code example below shows what (not) to do.

Bad: dashboard>screen>~~row~~>section>chart (row-tag is missing)

```
<?xml version="1.0" encoding="UTF-8"?>
<dashboard>
  <screen>
        <section>
            <chart>
            </chart>
        </section>
</dashboard>
```

Good: dashboard>screen>row>section>chart

```
<?xml version="1.0" encoding="UTF-8"?>
<dashboard>
  <screen>
      <row>
        <section>
            <chart>
            </chart>
        </section>
      </row>
</dashboard>
```

1.4.2 Evening out the width of section tags

As mentioned earlier, it is possible for the user to set the width of the `section` - tags themselves. However, you will notice that if you declare a witdh smaller than 100%, the chart will not be aligned quite nicely. The software arranges the sections horizontally from left to right in each row; In case the sum of all sections widths in a row isn't equal to 100, a strange layout may be the result. Therefore it is advised you always declare sections untill the total sum equals 100, even if you're not planning on filling in all sections with charts. Exceeding this total will result in unpredictable layout arrangements.

2. The .csv file - data

The datafile is often easy to export from Excel, but there are still a few things that should be checked beforehand to avoid reading errors.

2.1 Best Practices

2.1.1 Comma Separated Values (but not quite)

The software is setup to read csv-files where the separator isn't a comma (,) but a semicolon (;). Loading in a .csv-file that uses commas to split colomns will result in an error.

2.1.2 Add a semicolon (;) to the end of each row

To guarantee smooth sailing, a semicolon is to be put at the end of each row.

3. Deploying the dashboard

Now that both the config file and the data file are set up, the dashboard can be deployed in a few simple steps.

1. Open the index.html file you'll find in this folder. This should open up a (local) webpage. On this page, you'll see two upload buttons: One for the .xml-file (structure), another for the .csv-file (data)
2. Upload the config.xml file (scroll down to see the changes)
3. Upload the data.csv file (scroll down to see the changes)

Everything should be up and running now!