Configuration Guide - Dashboard Testproject

by Hube Van Loey

- 1. The config.xml file
  - 1.1 Setting up the config.xml file
  - 1.2 Structure Tags
  - 1.3 Chart Tags
    - 1.3.2 Single Nightingale Chart
    - 1.3.2 Multiple Nightingale Chart
  - 1.4 Best Practices
    - 1.4.1 No gaps in hierarchical structure
    - 1.4.2 Evening out the width of section tags
- 2. The .csv file - data
  - 2.1 Best Practices
    - 2.1.1 Comma Separated Values (but not quite)
    - 2.1.2 Add a semicolon (;) to the end of each row
- 3. Deploying the dashboard

This Dashboard Testproject Configuration 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:

- `<?xml?>` - tag
  This tag is used to specify the xml version and the encoding of the file.
- `<dashboard>` - tag
  This tag is used to initialize the whole dashboard. Everything within these tags will be visible on the dashboard.

  After adding these tags to your config.xml-file, it should look something like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<dashboard>
  <!-- everything for your dashboard comes here! -->
</dashboard>
```

1.2 Structure Tags

Now that we have declared our dashboard, 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. `<screen>` - tag
   This 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.
2. `<row>` - tag
   Each screen can be divided into different rows. The software automatically ensures that these rows are equal in height regardless of the number. If only one row is used, it will be the size of a `screen` tag.
3. `<section></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 can set the width of these sections yourself using the `width` attribute. Accepted values are:
   - `width="1/1"` (100% screen width)
   - `width="3/4"` (75% screen width)
   - `width="2/3"` (66% screen width)
   - `width="1/2"` (50% screen width)
   - `width="1/3"` (33% screen width)
   - `width="1/4"` (25% screen width)

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

1.3 Chart Tags

Once the structure of the dashboard is set up, the 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:

- `title=""` (optional)
  With this attribute, the chart title can be declared.

- `subtitle=""` (optional)
  With this attribute, the chart subtitle can be declared.

1.3.2 Single Nightingale Chart

- `type="sntgchart"` (required)
  With this attribute, the chart type is declared. This way, the software knows how to set up this specific chart.

- `targetColumn="nameOfTargetColumn"` (required)
  With this attribute, the column in the csv from which you want to visualize data is declared.

1.3.2 Multiple Nightingale Chart

- `type="mntgchart"` (required)
  With this attribute, the chart type is declared. This way, the software knows how to set up this specific chart.

- `timeColumn="nameOfTimeColumn"` (required)
  With this attribute, the column in the csv on which you base the time factor is declared.

- `targetColumn="nameOfTargetColumn"` (required)
  With this attribute, the column in the csv from which you want to visualize data (based on the time factor) is declared.

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.

Good: dashboard>screen>row>section>chart

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

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

```
<?xml version="1.0" encoding="UTF-8"?>
<dashboard>
  <screen>
        <section>
            <chart>
            </chart>
        </section>
</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. *Important note: This is something that occured during bug-fixing; Not really sure wether or not it's actually necessary*

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.
2. Upload the config.xml file
3. Upload the data.csv file

Everything should be up and running now!