

Generating HTML Pages with Highcharts using Groovy and Velocity

The Highcharts Javascript library is a sophisticated library that renders charts, which can be embedded and displayed in HTML pages. There are several possibilities of different chart types – also combined ones – and in the web browser and the charts are dynamic, meaning you can click to hide certain data, tooltips are shown and also drill-down is possible. For more information and demos goto <http://www.highcharts.com>.

Groovy is a programming language which is build on top of Java but enhances the Java language with many features that make it interesting, e.g. in the area of scripting. Groovy helps being “dynamic” (and “groovy” of course...).

Apache Velocity is a template engine, which allows to separate code from its representation, so that a programmer can do the technical work and e.g. a Web-designer can do the layout work (velocity templates) to create HTML pages. Basically a template contains placeholders where the data will go.

Combining these three tools, one gets a clear, fast and well-designed process to create HTML pages containing Highcharts charts, which can also be produced in batch mode. A Groovy script will get the data for the chart(s) from e.g. a database and prepares it so that it fits the charting needs. Then the code will be merged with a velocity template and the result is a HTML page which displays the required charts.

There is a little Java library available, that is also used in this context. It is not more than some Java helper classes, which make the handling of the data easier: it is not always easy to write an SQL query that provides the data exactly in the way that Highcharts needs it. The helper classes provide an easy way to circumvent this.

Example groovy script:

The script below is commented so that most of what happens should be clear:

```
// import statements for Java classes that are required
import groovy.sql.Sql
import org.apache.velocity.Template
import org.apache.velocity.VelocityContext
import org.apache.velocity.app.VelocityEngine
import java.text.SimpleDateFormat
// the following import statements are for the helper library
import com.datamelt.highcharts.Chart;
import com.datamelt.highcharts.Constants;

//calculate today's date's year
today = new Date()
sdf = new SimpleDateFormat("yyyy")
todayYear = sdf.format(today)

// these value are required so that velocity finds the template
RESOURCE_PATH = "file.resource.loader.path";
templateFolder="/home/uwe/development/velocity_templates"
templateName="hc_05.vm"

// the above resources are put in a properties class
Properties properties = new Properties()
properties.setProperty(RESOURCE_PATH, templateFolder)
// create a velocity engine object
VelocityEngine ve = new VelocityEngine()
// initialize it with the properties defined above
ve.init(properties)
// get the template
Template t = ve.getTemplate(templateName)
// we will use a StringWriter object which will contain the results
StringWriter writer = new StringWriter()
// define a SQL connection
def sql = Sql.newInstance("jdbc:mysql://localhost/<dbname>", "<username>","<password>",
"com.mysql.jdbc.Driver")
// define the SQL query to execute
sqlQuery="select month,company, value from staging_final where year like
date_format(date_add(curdate(),interval -1 year),'%Y') and station='xxx' and
kpi_type='Working Hours' group by month, company order by month,company"

// the following an object from the helper library
// there are two parameters passed to the class: the title and subtitle of
// the chart
Chart chart = new Chart("KPI Report","Working hours " + todayYear)

// loop over the results (records) which were return by the query
//
// the data is return in three columns: month, company and value
// the first contains the calendar month the data corresponds to
// the second is the name of the company that the data for the month applies to
// the third column contains the value
//
// example:
//
// month      company      value
// =====
// 1          Company A    1000
// 1          Company B    1100
// 2          Company A    2000
// 2          Company B    2300
//
// loop over all records that are returned by the query
sql.eachRow(sqlQuery){ row →
    // add the data to the chart object - first parameter is the category
    // in a highcharts chart corresponding to our first column "month"
    // the second parameter is the series in highcharts, which corresponds
    // to our field "company" and the third parameter is the value.
    chart.addData(row.month.toString(), row.company,row.value)
}

// create a velocity context
VelocityContext context = new VelocityContext()
// put the object in the context that are required in the velocity template
// put the chart object
```

Generating HTML Pages with Highcharts using Groovy and Velocity

```
context.put("chart",chart)
// put a list of all month of the year (Jan, Feb, Mar, Apr,...)
context.put("allmonths",Constants.MONTH_NAMES);

// now merge the context with the template
t.merge(context,writer)
// output the result to the console
println writer.toString()
```

The chart object in the groovy script collects all the data that will be required to display a proper chart using Highcharts. It offers convenient methods to loop over the series of data and to retrieve the value in a format appropriate for highcharts. The output here is going to the console, but you could also change the code to write it to a file or send it back to a client which uses a web browser.

Example Velocity template:

Basically a Velocity template - in this case - is a HTML page with placeholders for the data or other information we have generated in the groovy script above. A placeholder starts with a dollar sign (\$) and usually is in brackets:

```
<!DOCTYPE HTML>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <title>testchart</title>

    <style type="text/css">
      body { background-color:white; color:#FFFFFF; }
      a:link { color:#FF9966; }
      a:visited { color:#FF9900; }
      a:active { color:#FFFFFF; }
    </style>
    <!-- 1. Add these JavaScript inclusions in the head of your page -->
    <script type="text/javascript" src="js/jquery/jquery-1.6.4.min.js"></script>
    <script type="text/javascript" src="js/highcharts/js/highcharts.js"></script>
    <script type="text/javascript" src="js/themes/grid.js"></script>
    <script type="text/javascript" src="js/modules/exporting.js"></script>

    <!-- 2. Add the JavaScript to initialize the chart on document ready -->
    <script type="text/javascript">
var chart;
$(document).ready(function() {
  chart = new Highcharts.Chart({
    chart: {
      renderTo: 'container',
      defaultSeriesType: 'line'
    },
    title: {
      text: '${chart.title}'
    },
    subtitle: {
      text: '${chart.subtitle}'
    },
    xAxis: {
      categories: [${allmonths}],
      title: {
        text: null
      }
    },
    yAxis: {
      min: 0,
      title: {
        text: 'Working Hours',
        align: 'middle'
      }
    },
    exporting: {
      enabled: true
    },
    tooltip: {
      formatter: function() {
        return ''+

```

Generating HTML Pages with Highcharts using Groovy and Velocity

```
                this.series.name + ': '+ this.y + ' hours';
            }
        },
        plotOptions: {
            bar: {
                dataLabels: {
                    enabled: true
                }
            }
        },
        legend: {
            layout: 'vertical',
            align: 'right',
            verticalAlign: 'top',
            y: 10,
            floating: true,
            borderWidth: 0,
            backgroundColor: '#FFFFFF',
            shadow: true
        },
        credits: {
            enabled: false
        },
        series: [
            #foreach($series in ${chart.getSeries()})
                { name: ${series.nameQuoted}, data: [${series.values}]},
            #end
        ]
    });
});
</script>
</head>
<body>
    <div id="container" style="width: 100%; height: 500px; margin: 10 auto"></div>
</body>
</html>
```

Most of the code is just HTML but the interesting part of the Velocity template is this part:

```
xAxis: {
    categories: [${allmonths}],
    title: {
        text: null
    }
}
```

Here the categories are set. In our example we use all 12 month as the categories, because all month should be displayed (if we have values or not). They come from the "allmonth" variable which is a constant value from the helper library. These values will go into the x-axis of the chart.

And this part:

```
series: [
    #foreach($series in ${chart.getSeries()})
        { name: ${series.nameQuoted}, data: [${series.values}]},
    #end
]
```

The code of the helper library allows you to retrieve the series of data that originally came from the SQL query. The "foreach" statement loops over all series and inserts the name and the values of the series.

This is all. The Groovy script above – once run – will retrieve the data and merge it with the template. This means that the placeholders in the template are replaced with the real values of the object referenced that come from the groovy script. The result is a HTML page ready to be displayed in the web browser. This can be a local file but the HTML code/file can also be dynamically generated on a server.

Note that you need the highcharts Javascript library and the jquery Javascript library to use

highcharts. They are referenced at the top of the HTML page and can be downloaded from the Internet.

Here is the produced HTML code:

When the groovy script finishes, it produces following output:

```
<!DOCTYPE HTML>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <title>testchart</title>

    <style type="text/css">
      body { background-color:white; color:#FFFFFF; }
      a:link { color:#FF9966; }
      a:visited { color:#FF9900; }
      a:active { color:#FFFFFF; }
    </style>

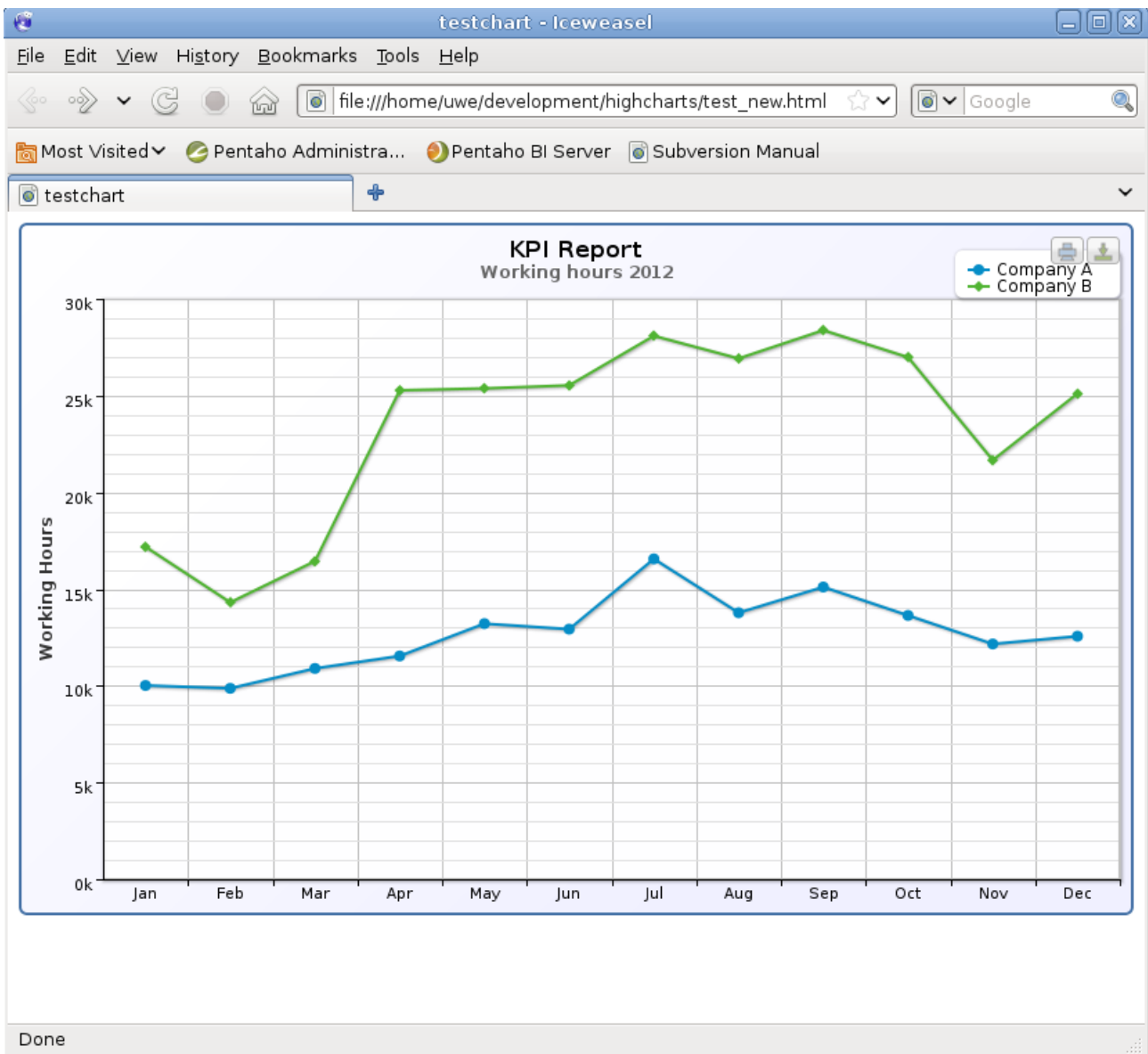
    <!-- 1. Add these JavaScript inclusions in the head of your page -->
    <script type="text/javascript" src="js/jquery-1.6.4.min.js"></script>
    <script type="text/javascript" src="js/highcharts.js"></script>
    <script type="text/javascript" src="js/themes/grid.js"></script>
    <script type="text/javascript" src="js/modules/exporting.js"></script>

    <!-- 2. Add the JavaScript to initialize the chart on document ready -->
    <script type="text/javascript">
var chart;
$(document).ready(function() {
  chart = new Highcharts.Chart({
    chart: {
      renderTo: 'container',
      defaultSeriesType: 'line'
    },
    title: {
      text: 'KPI Report'
    },
    subtitle: {
      text: 'Working hours 2012'
    },
    xAxis: {
      categories:
['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec'],
      title: {
        text: null
      }
    },
    yAxis: {
      min: 0,
      title: {
        text: 'Working Hours',
        align: 'middle'
      }
    },
    exporting: {
      enabled: true
    },
    tooltip: {
      formatter: function() {
        return '+'
          this.series.name + ': ' + this.y + ' hours';
      }
    },
    plotOptions: {
      bar: {
        dataLabels: {
          enabled: true
        }
      }
    },
    legend: {
      layout: 'vertical',
      align: 'right',
      verticalAlign: 'top',
```

Generating HTML Pages with Highcharts using Groovy and Velocity

```
        y: 10,
        floating: true,
        borderWidth: 0,
        backgroundColor: '#FFFFFF',
        shadow: true
    },
    credits: {
        enabled: false
    },
    series: [
        { name: 'Company A', data:
[10017.8,9868.25,10898.9,11542.0,13213.9,12928.5,16560.4,13778.3,15106.5,13637.3,12162.4,1256
3.4]},
        { name: 'Company B', data:
[17185.3,14312.8,16444.8,25274.9,25373.3,25531.3,28089.5,26920.5,28379.2,26983.6,21666.2,2510
7.1]},
    ]
    });
});
</script>
</head>
<body>
    <div id="container" style="width: 100%; height: 500px; margin: 10 auto"></div>
</body>
</html>
```

And finally here is an example chart generated with the groovy script and the Velocity template:



Summary:

The combination of the presented tools can help generate dynamic HTML code displaying Highcharts. Groovy provides a servlet that can be integrated into a web server. This way you can serve the Highcharts through the web server.

The example given is just a small part of what is possible by combining the tools. The user get a very flexible system to quickly produce code that is – through the use of the Velocity template engine – also easy to maintain and scalable.
