Grails Google Visualization Plugin - Reference Documentation

Benjamin Muschko

Version 2.3-SNAPSHOT

Table of Contents

1. Introduction 1
   1.1. Features 1
   1.2. Issues 1
   1.3. Release Notes 1
   1.4. Acknowledgments 4
   1.5. License 4
2. Usage 5
   2.1. Installation 5
   2.2. Configuration 5
   2.3. Basic usage 5
   2.4. Examples 6
   2.5. Events 7
   2.6. Formatters 8
3. Charts 10
   3.1. Pie Chart 10
   3.2. Bar Chart 10
   3.3. Bubble Chart 11
   3.4. Column Chart 12
   3.5. Area Chart 12
   3.6. Line Chart 13
   3.7. Scatter Chart 14
   3.8. Stepped Area Chart 14
   3.9. Candlestick Chart 15
   3.10. Combo Chart 15
   3.11. Gauge 16
   3.12. Table 16
   3.13. Map 17
   3.14. Annotated Time Line 17
   3.15. Annotation Chart 18
   3.16. Org Chart 19
   3.17. Intensity Map 19
   3.18. Geo Map 20
   3.19. Geo Chart 20
   3.20. Motion Chart 21
   3.21. Tree Map 21
   3.22. Timeline 22
   3.23. Calendar Chart 23
   3.24. Data Table Roles 23

4. Visualizations 25
4.1. Available Visualizations 25

Chapter 1. Introduction

This plugin provides a taglib for the interactive charts of the Google Visualization API.

In addition to this document, you may want to read the Google Visualization documentation here.

1.1. Features

- Supports the following visualizations: Annotated Time Line, Area Chart, Bar Chart, Bubble Chart, Candlestick Chart, Column Chart, Combo Chart, Gauge, Geo Chart Geo Map, Intensity Map, Line Chart, Map, Motion Chart, Organizational Chart, Pie Chart, Scatter Chart, Stepped Area Chart, Table Chart, Time Line Chart and Tree Map. See the gallery for more information.
- Implements redesigned charts (Area, Bar, Bubble, Candlestick, Column, Combo, Line, Pie, Scatter and Stepped Area Charts) from the previously known as "Core Chart package" as well as the deprecated versions.
- Provides implementations for table formatters TableArrowFormat, TableBarFormat, TableColorFormat, TableDateFormat, TableNumberFormat and TablePatternFormat.
- Visualization Event Handling.
- Provides support for Data Tables Roles (since version 1.0)

1.2. Issues

The old JIRA links for Grails are not working anymore, so please ignore the issues links in documentation. Also, all issues must be reported to GitHub Issues.

1.3. Release Notes

October 09, 2015 (version 2.0)

- Provides support for Grails 3.x

January 07, 2015 (version 1.0.1)

- Added "title" config option to Calendar chart. Although it does not appear in the API documentation, the examples show that it is available.

January 04, 2015 (version 1.0)

- Updates the configuration options to the current Google Charts API. Important! This might introduce some backward compatibility issues as some parameters have been removed, replaced or have changed type. Eg: tooltipText and tooltipTextStyle replaced by tooltip of type object, ... (This will be noted in the README.md file and the changelog wiki page)
- Removed support for Image Charts as they are no longer part of the Google Charts API
- Now the object type can be declared using the Map notation as the preferred option instead of using an Expando object declaration. It still can be used though to avoid backward compatibility issues. Example: `legend="${[position: 'top', alignment: 'center']}"` instead of: `legend="${new Expando(position: 'top', alignment: 'center')}"`

- Replaced scriptlets with EL expressions in javascript template.
- Fixes issue #15
- Other minor fixes and tasks

December 11, 2014 (version 0.7.2)

- Fixes bug #32: the "error" event handling should be declared before drawing the chart and not after.

November 28, 2014 (version 0.7.1)

- Fixes bugs #28 and #24: Grails 2.3+ XSS prevention mechanism was HTML encoding the javascript output of the taglib and therefore breaking it. There is also a reference in JIRA: [GPGOOGLEVISUALIZATIONAPI-14])

April 30, 2014 (version 0.7)

- Fix ObjectRenderer to properly filter class and metaClass properties (see pull request #23).
- Add support for Timeline and Calendar charts (see pull request #22).

May 9, 2013 (version 0.6.2)

- Enable use of candlestick options (see pull request 12).

February 3, 2013 (version 0.6.1)

- Fixed HTML codec issues (see pull request 11).

November 11, 2012 (version 0.6)

- Internal cleanup (see pull request 10).

October 14, 2012 (version 0.5.6)

- Added missing bar attribute to several charts.

August 18, 2012 (version 0.5.5)

- Support forceIFrame parameter(see GPGOOGLEVISUALIZATIONAPI-12).

August 12, 2012 (version 0.5.4)

- Support vAxes parameter (see GPGOOGLEVISUALIZATIONAPI-11).

July 8, 2012 (version 0.5.3)

- Support for pattern property in NumberFormatter (see GPGOOGLEVISUALIZATIONAPI-10).

May 26, 2012 (version 0.5.2)

- Added isStacked attribute for ComboCoreChart.

May 17, 2012 (version 0.5.1)

- Provided support for marker style in GeoChart (see GPGOOGLEVISUALIZATIONAPI-7)
- Upgrade to Grails 2.0.1 and Release plugin 2.0.0.

January 29, 2012 (version 0.5)

- Added new visualizations: Bubble Chart, Stepped Area Chart.
- Allow all visualizations to use String data type for the attributes height and width (see GPGOOGLEVISUALIZATIONAPI-5).
- Allow all visualizations to use formatters (see GPGOOGLEVISUALIZATIONAPI-6).
- Provided taglib for Google JavaScript API import.

December 3, 2011 (version 0.4.3)

- Ready event has to be defined before calling draw on visualization (see GPGOOGLEVISUALIZATIONAPI-3).

October 2, 2011 (version 0.4.2)

- Ready event has to be defined before calling draw on visualization (see GPGOOGLEVISUALIZATIONAPI-2).

August 27, 2011 (version 0.4.1)

- Updated core chart parameters (see Google's release notes).

May 20, 2011 (version 0.4)

- Added new visualizations: Geo Chart, Candlestick Chart and Combo Chart.
- Exposed visualization parameter to set API version.

May 17, 2011 (version 0.3.1)

- Updated core chart parameters (see Google's release notes).

March 20, 2011 (version 0.3)

- Updated core chart parameters (see Google's release notes).
- Added Candlestick image chart.

November 10, 2010 (version 0.2.4)

- Updated core chart parameters (see Google's release notes).
- Improved String data type handling (see GPGOOGLEVISUALIZATIONAPI-1).

August 26, 2010 (version 0.2.3)

- Updated core chart parameters (see Google's release notes).

August 4, 2010 (version 0.2.2)

- Exposed visualization data JavaScript object (google.visualization.DataTable).
- Bugfixes: String parameters and cell labels needed to be escaped.

July 8, 2010 (version 0.2.1)

- Added onmouseover/onmouseout events to treemap.
- Support for dynamic loading.

June 5, 2010 (version 0.2)

- Added new visualization Tree Map.
- Added image charts: Pie Chart, Bar Chart, Area Chart, Line Chart and Sparkline.
- Map now provides the Terrain type.
- Support for cell object.

May 27, 2010 (version 0.1)

- Initial version.

1.4. Acknowledgments

Many thanks to all the users who reported issues and sent pull requests.

1.4.1. Authors and Contributors

- Benjamin Muschko
- Angel Ruiz
- Mansi Arora
- Robert Oschwald
- Uday Pratap Singh
- Tom Crossland
- Sdonyh
- Ryan
- Lewis Norton
- Zoran Stojakovic
- Puneet Behl
- Vmille

1.5. License

This plugin is released under the Apache License, Version 2.0

Chapter 2. Usage

2.1. Installation

2.1.1. For Grails 3.x

Add the following dependency under `build.gradle`:

```
compile "org.grails.plugins:grails-google-visualization:2.0"
```

2.1.2. For Grails 2.x

Add the following plugin under plugins in BuildConfig.groovy:

```
compile ":google-visualization:1.0.1"
```

2.2. Configuration

To get started using the Google Maps JavaScript API, you need to define an API key in the application configuration as follows:

application.yml

```
google:
    maps:
        key: YOUR_API_KEY
```

2.2.1. Get a Key/Authentication

All Google Maps JavaScript API applications require authentication. Please click here to read about authentication/key or to get an API key.

2.3. Basic usage

- The page you want to use the visualization in has to import the Google visualization API JavaScript library. You can do so by using the taglib `<gvisualization:apiImport/>` or by importing it using the HTML script tag.

```
<script type="text/javascript" src="https://www.google.com/jsapi"></script>
```

- All visualizations in the taglib use the namespace **gvisualization**.
- Apart from the configuration options specific to a visualization (see visualization specifications)

there are multiple attributes that you have to set for your visualization.

- ***name*** (optional) - JavaScript variable name for visualization object (defaults to "visualization").
- ***version*** (optional) - API version for visualization object (defaults to "1").
- ***elementId*** (required) - HTML div ID used to render visualization.
- ***dynamicLoading*** (optional) - Renders visualization over dynamic loading - required when rendered in AJAX response (defaults to false).
- ***language*** (optional) - Forces localized version of visualization. The language property is a two-letter ISO 639-1 language code.
- ***columns*** (required) - List of column data types and names to be displayed in the visualization.
- ***data*** (required) - List of data to be displayed for columns.
  - To express objects you use the Map notation inside an EL expression. Eg: [source,groovy] ---- legend="${[position: 'top', alignment: 'center']}" ----
  - If your data requires the usage of the cell object you can import and populate the class `org.grails.plugins.google.visualization.data.Cell`.

2.4. Examples

2.4.1. Rendering a visualization in a GSP

```
<%
def myDailyActivitiesColumns = [['string', 'Task'], ['number', 'Hours per Day']]
def myDailyActivitiesData = [['Work', 11], ['Eat', 2], ['Commute', 2], ['Watch TV',
2], ['Sleep', 7]]
%>

<gvisualization:pieCoreChart elementId="piechart" title="My Daily Activities"
width="${450}" height="${300}" columns="${myDailyActivitiesColumns}"
data="${myDailyActivitiesData}" />

<div id="piechart"></div>
```

2.4.2. Rendering a visualization in a AJAX response

GSP (`grails-app/views/index.gsp`)

```
<input type="button" value="Render Pie Chart"
onclick="${remoteFunction(controller:'visualization',action:'render',update:'chart')}"
>
<br>
<div id="chart"></div>
```

Controller (grails-app/controllers/VisualizationController.groovy)

```
class VisualizationController {
  def render = {
    def myDailyActivitiesColumns = [['string', 'Task'], ['number', 'Hours per Day']]
    def myDailyActivitiesData = [['Work', 11], ['Eat', 2], ['Commute', 2], ['Watch
TV', 2], ['Sleep', 7]]
    render template: "chart", model: ["myDailyActivitiesColumns":
myDailyActivitiesColumns,
    "myDailyActivitiesData": myDailyActivitiesData]
  }
}
```

Template (grails-app/views/visualization/_chart.gsp)

```
<div id="piechart"></div>
<gvisualization:pieCoreChart dynamicLoading="${true}" elementId="piechart" title="My
Daily Activities"
width="${450}" height="${300}" columns="${myDailyActivitiesColumns}"
data="${myDailyActivitiesData}"/>
```

2.5. Events

If you want to register an event handler for your visualization you can by adding an event attribute. The value you give that attribute is the name of the JavaScript function acting as callback handler. Please check the visualization specification for available event names.

The variable name of the visualization JavaScript object by default is "**visualization**", the name of the `google.visualization.DataTable` object is "**visualization_data**". You can always change the names by setting the taglib attribute **"name"**.

Example:

```
<%
def employeeColumns = [['string', 'Name'], ['string', 'Salary'], ['boolean', 'Full
Time Employee']]
def employeeData = [['Mike', '$10,000', true], ['Jim', '$8,000', false], ['Alice',
'$12,500', true],
['Bob', '$7,000', true]]
%>

<script type="text/javascript">
  function selectHandler(e) {
    alert('A table row was selected');
  }
</script>

<gvisualization:table elementId="table" width="${400}" height="${130}"
columns="${employeeColumns}"
data="${employeeData}" select="selectHandler" />
<div id="table"></div>
```

Since the **Table visualization** has both a "page" config option and a "page" event, to use the latest the taglib attribute is called **"page-event"**.

2.6. Formatters

You can apply formatters to all visualizations using the "formatters" taglib attribute as the underlying implementation is a google.visualization.DataTable.

However, using formatters makes the most sense for the Table visualization.

A full set of examples can be found on this GSP page.The value you have to pass in is a list of classes implementing the `org.grails.plugins.google.visualization.formatter.Formatter` interface. The implementations you can apply are the following:

```
org.grails.plugins.google.visualization.formatter.PatternFormatter
org.grails.plugins.google.visualization.formatter.NumberFormatter
org.grails.plugins.google.visualization.formatter.DateFormatter
org.grails.plugins.google.visualization.formatter.ColorRange
org.grails.plugins.google.visualization.formatter.ColorFormatter
org.grails.plugins.google.visualization.formatter.BarFormatter
org.grails.plugins.google.visualization.formatter.ArrowFormatter
```

Example:

```
<%@ page import="org.grails.plugins.google.visualization.formatter.BarFormatter" %>
<%
def departmentRevenueColumns = [['string', 'Department'], ['number', 'Revenues']]
def departmentRevenueData = [['Shoes', 10700], ['Sports', -15400], ['Toys', 12500],
['Electronics', -2100],
['Food', 22600], ['Art', 1100]]
def barFormatter = new BarFormatter(1)
barFormatter.width = 120
def barFormatters = [barFormatter]
%>

<gvisualization:table elementId="barformat_div" allowHtml="${true}"
showRowNumber="${true}"
columns="${departmentRevenueColumns}" data="${departmentRevenueData}"
formatters="${barFormatters}"/>
<div id="barformat_div"></div>
```

Chapter 3. Charts

The Google Maps api that Google Charts uses to do geocoding of named locations has been updated several months ago (not sure how long) and the old uses appear to have been 'grandfathered' in without complaint. So, going forward we need to define API key as shown below while rendering Maps, GeoCharts etc.

application.yml

```
google:
    maps:
        key: YOUR_API_KEY
```

To get a API key click here

3.1. Pie Chart

VisualizationController.groovy

```
class VisualizationController {
  def pieChart() {
   List myDailyActivitiesColumns = [['string', 'Task'], ['number', 'Hours per Day']]
   List myDailyActivitiesData = [['Work', 11], ['Eat', 2], ['Commute', 2], ['Watch
TV', 2], ['Sleep', 7]]
   render template: "pieChart", model: ["myDailyActivitiesColumns":
myDailyActivitiesColumns,
    "myDailyActivitiesData": myDailyActivitiesData]
  }
}
```

_pieChart.gsp

```
<gvisualization:pieCoreChart elementId="piechart" title="My Daily Activities"
width="${450}" height="${300}" columns="${myDailyActivitiesColumns}"
data="${myDailyActivitiesData}" />


<div id="piechart"></div>
```

3.2. Bar Chart

VisualizationController.groovy

```
class VisualizationController {
  def barChart() {
    List companyPerformanceColumns = [['string', 'Year'], ['number', 'Sales'],
['number', 'Expenses']]
    List companyPerformanceData = [['2004', 1000, 400], ['2005', 1170, 460], ['2006',
660, 1120], ['2007', 1030, 540]]
    render template: "barChart", model: ["companyPerformanceColumns":
companyPerformanceColumns, "companyPerformanceData": companyPerformanceData]
  }
}
```

_barChart.gsp

```
 <gvisualization:barCoreChart elementId="barchart" title="Company Performance"
width="${400}" height="${240}" vAxis="${[title: 'Year', titleColor: 'red']}"
columns="${companyPerformanceColumns}" data="${companyPerformanceData}" />


 <div id="barchart"></div>
```

3.3. Bubble Chart

VisualizationController.groovy

```
class VisualizationController {
  def bubbleChart() {
    List lifeExpectancyFertilityRateColumns =[['string', 'ID'], ['number', 'Life
Expectancy'], ['number', 'Fertility Rate'], ['string', 'Region'], ['number',
'Population']]
    List lifeExpectancyFertilityRateData = [['CAN', 80.66, 1.67, 'North America',
33739900], ['DEU', 79.84, 1.36, 'Europe', 81902307], ['DNK', 78.6, 1.84, 'Europe',
5523095], ['EGY', 72.73, 2.78, 'Middle East', 79716203], ['GBR', 80.05, 2, 'Europe',
61801570], ['IRN', 72.49, 1.7, 'Middle East', 73137148], ['IRQ', 68.09, 4.77, 'Middle
East', 31090763], ['ISR', 81.55, 2.96, 'Middle East', 7485600], ['RUS', 68.6, 1.54,
'Europe', 141850000], ['USA', 78.09, 2.05, 'North America', 307007000]]
    render template: "bubbleChart", model:
["lifeExpectancyFertilityRateColumns":lifeExpectancyFertilityRateColumns,
"lifeExpectancyFertilityRateData": lifeExpectancyFertilityRateData]
  }
}
```

_bubbleChart.gsp

```
<gvisualization:bubbleCoreChart elementId="bubblechart" title="Correlation between
life expectancy, fertility rate and population of some world countries (2010)"
hAxis="${[title: 'Life Expectancy']}" vAxis="${[title: 'Fertility Rate']}"
bubble="${[textStyle: '{fontSize: 11}']}"
columns="${lifeExpectancyFertilityRateColumns}"
data="${lifeExpectancyFertilityRateData}" />


<div id="bubblechart" style="width: 900px; height: 500px;"></div>
```

3.4. Column Chart

VisualizationController.groovy

```
class VisualizationController {
  def columnChart() {
    List companyPerformanceColumns = [['string', 'Year'], ['number', 'Sales'],
['number', 'Expenses']]
    List companyPerformanceData = [['2004', 1000, 400], ['2005', 1170, 460], ['2006',
660, 1120], ['2007', 1030, 540]]
    render template: "columnChart", model:
["companyPerformanceColumns":companyPerformanceColumns, "companyPerformanceData":
companyPerformanceData]
  }
}
```

_columnChart.gsp

```
<gvisualization:columnCoreChart elementId="columnchart" title="Company Performance"
width="${400}" height="${240}" hAxis="${[title: 'Year', titleColor: 'red']}"
columns="${companyPerformanceColumns}" data="${companyPerformanceData}" />


<div id="columnchart"></div>
```

3.5. Area Chart

VisualizationController.groovy

```
class VisualizationController {
  def areaChart() {
    List companyPerformanceColumns = [['string', 'Year'], ['number', 'Sales'],
['number', 'Expenses']]
    List companyPerformanceData = [['2004', 1000, 400], ['2005', 1170, 460], ['2006',
660, 1120], ['2007', 1030, 540]]
    render template: "areaChart", model: ["companyPerformanceColumns",
companyPerformanceColumns, "companyPerformanceData": companyPerformanceData]
  }
}
```

_areaChart.gsp

```
<gvisualization:areaCoreChart elementId="areachart" title="Company Performance"
width="${400}" height="${240}" hAxis="${[title: 'Year', titleColor: 'red']}"
columns="${companyPerformanceColumns}" data="${companyPerformanceData}" />


<div id="areachart"></div>
```

3.6. Line Chart

VisualizationController.groovy

```
class VisualizationController {
  def lineChart() {
    List companyPerformanceColumns = [['string', 'Year'], ['number', 'Sales'],
['number', 'Expenses']]
    List companyPerformanceData = [['2004', 1000, 400], ['2005', 1170, 460], ['2006',
660, 1120], ['2007', 1030, 540]]
    render template: "lineChart", model: ["companyPerformanceData":
companyPerformanceData, "companyPerformanceColumns": companyPerformanceColumns]
  }
}
```

_lineChart.gsp

```
<gvisualization:lineCoreChart elementId="linechart" width="${400}" height="${240}"
title="Company Performance" columns="${companyPerformanceColumns}"
data="${companyPerformanceData}" />

<div id="linechart"></div>
```

3.7. Scatter Chart

VisualizationController.groovy

```
class VisualizationController {
  def scatterChart() {
    List weightByAgeColumns = [['number', 'Age'], ['number', 'Weight']]
    List weightByAgeData = [[8, 12], [4, 5.5], [11, 14], [4, 5], [3, 3.5], [6.5, 7]]
    render template: "scatterChart", model: ["weightByAgeData": weightByAgeData]
  }
}
```

_scatterChart.gsp

```
<gvisualization:scatterCoreChart elementId="scatterchart" width="${400}"
height="${240}" title="Age vs. Weight comparison" hAxis="${[title: 'Age', minValue: 0,
maxValue: 15]}" vAxis="${[title: 'Weight', minValue: 0, maxValue: 15]}" legend="none"
columns="${weightByAgeColumns}" data="${weightByAgeData}" />


<div id="scatterchart"></div>
```

3.8. Stepped Area Chart

VisualizationController.groovy

```
class VisualizationController {
  def steppedAreaChart() {
    List accumulatedRatingColumns = [['string', 'Director (Year)'], ['number', 'Rotten
Tomatoes'], ['number', 'IMDB']]
    List accumulatedRatingData = [['Alfred Hitchcock (1935)', 8.4, 7.9], ['Ralph
Thomas (1959)', 6.9, 6.5], ['Don Sharp (1978)',6.5, 6.4], ['James Hawes (2008)', 4.4,
6.2]]
    render template: "steppedAreaChart", model: ["accumulatedRatingColumns":
accumulatedRatingColumns, "accumulatedRatingData":accumulatedRatingData]
  }
}
```

_steppedAreaChart.gsp

```
<gvisualization:steppedAreaCoreChart elementId="steppedareachart" width="${400}"
height="${240}" title="The decline of \'The 39 Steps\'" vAxis="${[title: 'Accumulated
Rating']}" isStacked="${true}" columns="${accumulatedRatingColumns}"
data="${accumulatedRatingData}" />


<div id="steppedareachart"></div>
```

3.9. Candlestick Chart

VisualizationController.groovy

```
class VisualizationController {
  def columnChart() {
    List countByDayColumns = [['string', 'Day'], ['number', ''], ['number', ''],
['number', ''], ['number', '']]
    List countByDayData = [['Mon', 20, 28, 38, 45], ['Tues', 31, 38, 55, 66], ['Wed',
50, 55, 77, 80], ['Thurs', 50, 77, 66, 77], ['Fri', 15, 66, 22, 68]]
    Map candlestickOptions = [hollowIsRising: true]
    render template: "candlestickChart", model: ["countByDayColumns":
countByDayColumns, "countByDayData": countByDayData, "candlestickOptions":
candlestickOptions]
  }
}
```

_candlestickChart.gsp

```
<gvisualization:candlestickCoreChart elementId="candlestickchart" legend="none"
columns="${countByDayColumns}" data="${countByDayData}"
candlestick="${candlestickOptions}" />


<div id="candlestickchart" style="width: 300px; height: 300px;"></div>
```

3.10. Combo Chart

VisualizationController.groovy

```
class VisualizationController {
  def comboChart() {
   List monthlyCoffeeProdByCountryColumns = [['string', 'Month'], ['number',
'Bolivia'], ['number', 'Ecuador'], ['number', 'Madagascar'], ['number', 'Papua
Guinea'], ['number', 'Rwanda'], ['number', 'Average']]
   List monthlyCoffeeProdByCountryData = [['2004/05', 165, 938, 522, 998, 450, 614.6],
['2005/06', 135, 1120, 599, 1268, 288, 682], ['2006/07', 157, 1167, 587, 807, 397,
623], ['2007/08', 139, 1110, 615, 968, 215, 609.4], ['2008/09', 136, 691, 629, 1026,
366, 569.6]]
   render template: "comboChart", model: ["monthlyCoffeeProdByCountryColumns":
monthlyCoffeeProdByCountryColumns, "monthlyCoffeeProdByCountryData":
monthlyCoffeeProdByCountryData]
  }
}
```

_comboChart.gsp

```
<gvisualization:comboCoreChart elementId="combochart" title="Monthly Coffee Production
by Country" vAxis="${[title: 'Cups']}" hAxis="${[title: 'Month']}" seriesType="bars"
series="${[5: [type: 'line']]}" columns="${monthlyCoffeeProdByCountryColumns}"
data="${monthlyCoffeeProdByCountryData}" />


<div id="combochart" style="width: 700px; height: 400px;"></div>
```

3.11. Gauge

VisualizationController.groovy

```
class VisualizationController {
  def gauge() {
   List systemPerformanceColumns = [['string', 'Label'], ['number', 'Value']]
   List systemPerformanceData = [['Memory', 80], ['CPU', 55], ['Network', 68]]
   render template: "gauge", model: ["systemPerformanceColumns":
systemPerformanceColumns, "systemPerformanceData": systemPerformanceData]
  }
}
```

_gauge.gsp

```
<gvisualization:gauge elementId="gauge" width="${400}" height="${120}" redFrom="${90}"
redTo="${100}" yellowFrom="${75}" yellowTo="${90}" minorTicks="${5}"
columns="${systemPerformanceColumns}" data="${systemPerformanceData}" />


<div id="gauge"></div>
```

3.12. Table

VisualizationController.groovy

```
class VisualizationController {
  def table() {
   List employeeColumns = [['string', 'Name'], ['string', 'Salary'], ['boolean', 'Full
Time Employee']]
   List employeeData = [['Mike', '$10,000', true], ['Jim', '$8,000', false], ['Alice',
'$12,500', true], ['Bob', '$7,000', true]]
   render template: "table", model: ["employeeColumns": employeeColumns,
"employeeData": employeeData]
  }
}
```

_table.gsp

```
<gvisualization:table elementId="table" width="${400}" height="${130}"
columns="${employeeColumns}" data="${employeeData}" select="selectHandler"
ready="readyHandler"/>


<div id="table"></div>
```

3.13. Map

VisualizationController.groovy

```
class VisualizationController {
  def map() {
   List mapColumns = [['number', 'Lat'], ['number', 'Lon'], ['string', 'Name']]
   List mapData = [[37.4232, -122.0853, 'Work'], [37.4289, -122.1697, 'University'],
[37.6153, -122.3900, 'Airport']
   render template: "map", model: ["mapColumns": mapColumns, "mapData": mapData]
  }
}
```

_map.gsp

```
<gvisualization:map elementId="map" columns="${mapColumns}" data="${mapData}" />


<div id="map" style="width: 400px; height: 300px"></div>
```

3.14. Annotated Time Line

VisualizationController.groovy

```
class VisualizationController {
  def annotatedTimeLine() {
   List pensColumns = [['date', 'Date'], ['number', 'Sold Pencils'], ['string',
'title1'], ['string', 'text1'], ['number', 'Sold Pens'], ['string', 'title2'],
['string', 'text2']]
   List pensData = [[DateUtil.createDate(2008, 1, 1), 30000, null, null, 40645, null,
null], [DateUtil.createDate(2008, 1, 2), 14045, null, null, 20374, null, null],
[DateUtil.createDate(2008, 1, 3), 55022, null, null, 50766, null, null],
[DateUtil.createDate(2008, 1, 4), 75284, null, null, 14334, 'Out of Stock','Ran out of
stock on pens at 4pm'], [DateUtil.createDate(2008, 1, 5), 41476, 'Bought Pens','Bought
200k pens', 66467, null, null], [DateUtil.createDate(2008, 1, 6), 33322, null, null,
39463, null, null]]
   render template: "annotatedTimeLine", model: ["pensColumns": pensColumns,
"pensData": pensData]
  }
}
```

_annotatedTimeLine.gsp

```
<gvisualization:annotatedTimeLine elementId="annotatedtimeline"
columns="${pensColumns}" data="${pensData}" />


<div id="annotatedtimeline" style='width: 700px; height: 240px;'></div>
```

3.15. Annotation Chart

VisualizationController.groovy

```
class VisualizationController {
  def annotationChart() {
   List pensColumns = [['date', 'Date'], ['number', 'Sold Pencils'], ['string',
'title1'], ['string', 'text1'], ['number', 'Sold Pens'], ['string', 'title2'],
['string', 'text2']]
   List pensData = [[DateUtil.createDate(2008, 1, 1), 30000, null, null, 40645, null,
null], [DateUtil.createDate(2008, 1, 2), 14045, null, null, 20374, null, null],
[DateUtil.createDate(2008, 1, 3), 55022, null, null, 50766, null, null],
[DateUtil.createDate(2008, 1, 4), 75284, null, null, 14334, 'Out of Stock','Ran out of
stock on pens at 4pm'], [DateUtil.createDate(2008, 1, 5), 41476, 'Bought Pens','Bought
200k pens', 66467, null, null], [DateUtil.createDate(2008, 1, 6), 33322, null, null,
39463, null, null]]
   render template: "annotationChart", model: ["pensColumns": pensColumns, "pensData":
pensData]
  }
}
```

_annotationChart.gsp

```
<gvisualization:annotationChart elementId="annotationchart" columns="${pensColumns}"
data="${pensData}" />


<div id="annotationchart" style='width: 700px; height: 240px;'></div>
```

3.16. Org Chart

VisualizationController.groovy

```
class VisualizationController {
  def orgChart() {
   def orgColumns = [['string', 'Name'], ['string', 'Manager'], ['string', 'ToolTip']]
   def orgData = [[new Cell(value: 'Mike', label: 'Mike<div style="color:red; font-
style:italic">President</div>'), '', 'The President'], [new Cell(value: 'Jim', label:
'Jim<div style="color:red; font-style:italic">Vice President<div>'), 'Mike', 'VP'],
['Alice', 'Mike', ''], ['Bob', 'Jim', 'Bob Sponge'], ['Carol', 'Bob', '']]
   render template: "orgChart", model: ["orgColumns": orgColumns, "orgData": orgData]
  }
}
```

_orgChart.gsp

```
<gvisualization:orgChart elementId="orgchart" allowHtml="${true}"
columns="${orgColumns}" data="${orgData}" />


<div id="orgchart"></div>
```

3.17. Intensity Map

VisualizationController.groovy

```
class VisualizationController {
  def intensityMap() {
   def popularityColumns = [['string', 'Country'], ['number', 'Popularity']]
   def popularityData = [['Germany', 200], ['United States', 300], ['Brazil', 400],
['Canada', 500], ['France', 600], ['RU', 700]]
   render template: "intensityMap", model: ["populationColumns": populationColumns,
"populationData": populationData]
  }
}
```

_intensityMap.gsp

```
<gvisualization:intensityMap elementId="intensitymap" columns="${populationColumns}"
data="${populationData}" />


<div id="intensitymap"></div>
```

3.18. Geo Map

VisualizationController.groovy

```
class VisualizationController {
  def geoMap() {
    def popularityColumns = [['string', 'Country'], ['number', 'Popularity']]
    def popularityData = [['Germany', 200], ['United States', 300], ['Brazil', 400],
['Canada', 500], ['France', 600], ['RU', 700]]
    render template: "geoMap", model: ["popularityColumns": popularityColumns,
"popularityData": popularityData]
  }
}
```

_geoMap.gsp

```
<gvisualization:geoMap elementId="geomap" columns="${popularityColumns}"
data="${popularityData}" />


<div id="geomap"></div>
```

3.19. Geo Chart

VisualizationController.groovy

```
class VisualizationController {
  def geoChart() {
    def popularityColumns = [['string', 'Country'], ['number', 'Popularity']]
def popularityData = [['Germany', 200], ['United States', 300], ['Brazil', 400],
['Canada', 500], ['France', 600], ['RU', 700]]
    render template: "geoChart", model: ["popularityColumns": popularityColumns,
"popularityData": popularityData]
  }
}
```

_geoChart.gsp

```
<gvisualization:geoChart elementId="geochart" width="${556}" height="${347}"
columns="${popularityColumns}" data="${popularityData}" />


<div id="geochart"></div>
```

3.20. Motion Chart

VisualizationController.groovy

```
class VisualizationController {
  def motionChart() {
   List fruitColumns = [['string', 'Fruit'], ['date', 'Date'], ['number', 'Sales'],
['number', 'Expenses'], ['string', 'Location']]
   List fruitData = [['Apples', DateUtil.createDate(1988, 0, 1), 1000, 300, 'East'],
['Oranges', DateUtil.createDate(1988, 0, 1), 1150, 200, 'West'], ['Bananas',
DateUtil.createDate(1988, 0, 1), 300, 250, 'West'], ['Apples',
DateUtil.createDate(1989, 6, 1), 1200, 400, 'East'], ['Oranges',
DateUtil.createDate(1989, 6, 1), 750, 150, 'West'], ['Bananas',
DateUtil.createDate(1989, 6, 1), 788, 617, 'West']]
   render template: "motionChart", model: ["fruitColumns": fruitColumns, "fruitData":
fruitData]
  }
}
```

_motionChart.gsp

```
<gvisualization:motionChart elementId="motionchart" columns="${fruitColumns}"
data="${fruitData}" />


<div id="motionchart"></div>
```

3.21. Tree Map

VisualizationController.groovy

```
class VisualizationController {
  def treeMap() {
   List marketByRegionColumns = [['string', 'Region'], ['string', 'Parent'],
['number', 'Market trade volume (size)'], ['number', 'Market increase/decrease
(color)']]
   List marketByRegionData = [['Global', null, 0, 0], ['America', 'Global', 0, 0],
['Europe', 'Global', 0, 0], ['Asia', 'Global', 0, 0], ['Australia', 'Global', 0, 0],
['Africa', 'Global', 0, 0], ['Brazil', 'America', 11, 10], ['USA', 'America', 52, 31],
['Mexico', 'America', 24, 12], ['Canada', 'America', 16, -23], ['France', 'Europe',
42, -11], ['Germany', 'Europe', 31, -2], ['Sweden', 'Europe', 22, -13], ['Italy',
'Europe', 17, 4], ['UK', 'Europe', 21, -5], ['China', 'Asia', 36, 4], ['Japan',
'Asia', 20, -12], ['India', 'Asia', 40, 63], ['Laos', 'Asia', 4, 34], ['Mongolia',
'Asia', 1, -5], ['Israel', 'Asia', 12, 24], ['Iran', 'Asia', 18, 13], ['Pakistan',
'Asia', 11, -52], ['Egypt', 'Africa', 21, 0], ['S. Africa', 'Africa', 30, 43],
['Sudan', 'Africa', 12, 2], ['Congo', 'Africa', 10, 12], ['Zair', 'Africa', 8, 10]]
   render template: "treeMap", model: ["marketByRegionColumns": marketByRegionColumns,
"marketByRegionData": marketByRegionData]
  }
}
```

_treeMap.gsp

```
<gvisualization:treeMap elementId="treemap" minColor="#f00" midColor="#ddd"
maxColor="#0d0" headerHeight="${15}" fontColor="black" showScale="${true}"
columns="${marketByRegionColumns}" data="${marketByRegionData}" />

<div id="treemap" style="width: 900px; height: 500px;"></div>
```

3.22. Timeline

VisualizationController.groovy

```
class VisualizationController {
  def timeline() {
   List timelineColumns = [['string', 'President'], ['date', 'Start'], ['date',
'End']]
   List timelineData = [['Washington',  DateUtil.createDate(1789, 3, 29),
DateUtil.createDate(1797, 2, 3)], ['Adams',  DateUtil.createDate(1797, 2, 3),
DateUtil.createDate(1801, 2, 3)], ['Jefferson',  DateUtil.createDate(1801, 2, 3),
DateUtil.createDate(1809, 2, 3)]]
   render template: "timeline", model: ["timelineColumns": timelineColumns,
"timelineData": timelineData]
  }
}
```

_timeline.gsp

```
<gvisualization:timeLine elementId="timeline" columns="${timelineColumns}"
data="${timelineData}" />


 <div id="timeline"></div>
```

3.23. Calendar Chart

VisualizationController.groovy

```
class VisualizationController {
  def calendarChart() {
   List calendarColumns = [['date', 'Date'], ['number', 'Won/Loss']]
   List calendarData = [[DateUtil.createDate(2012, 3, 13), 37032],
[DateUtil.createDate(2012, 3, 14), 38024], [DateUtil.createDate(2012, 3, 15), 38024],
[DateUtil.createDate(2012, 3, 16), 38108], [DateUtil.createDate(2012, 3, 17), 38229]]
   render template: "calendarChart", model: ["calendarColumns": calendarColumns,
"calendarData": calendarData]
  }
}
```

_calendarChart.gsp

```
<gvisualization:calendarChart elementId="calendarchart" columns="${calendarColumns}"
data="${calendarData}" />


<div id="calendarchart"></div>
```

3.24. Data Table Roles

VisualizationController.groovy

```
class VisualizationController {
  def dataTableRoles() {
   List dataTableRoleExampleColumns = [['string', 'Month'], ['number', 'Sales'],
[type: 'number', role: 'interval'], [type:'number', role:'interval'], [type:'string',
role:'annotation'], [type:'string', role:'annotationText'],
[type:'boolean',role:'certainty']]
   List dataTableRoleData = [
           ['April',1000,  900, 1100,  'A','Stolen data', true],
           ['May',  1170, 1000, 1200,  'B','Coffee spill', true],
           ['June',  660,  550,  800,  'C','Wumpus attack', true],
           ['July', 1030, null, null, null, null, false]
   ]
   render template: "dataTableRoles", model: ["dataTableRoleExampleColumns":
dataTableRoleExampleColumns, "dataTableRoleData": dataTableRoleData]
  }
}
```

_dataTableRoles.gsp

```
<gvisualization:lineCoreChart elementId="dataTableRoles" width="${400}"
height="${240}" columns="${dataTableRoleExampleColumns}" data="${dataTableRoleData}"
legend="${[position: 'top', alignment: 'center']}" />


<div id="dataTableRoles"></div>
```

Chapter 4. Visualizations

4.1. Available Visualizations

Working examples for all available visualization can be found on this GSP page to this page on GitHub.

4.1.1. Interactive Charts

- Pie Core Chart: `<gvisualization:pieCoreChart/>`
- Bar Core Chart: `<gvisualization:barCoreChart/>`
- Bubble Core Chart: `<gvisualization:bubbleCoreChart/>`
- Column Core Chart: `<gvisualization:columnCoreChart/>`
- Area Core Chart: `<gvisualization:areaCoreChart/>`
- Line Core Chart: `<gvisualization:lineCoreChart/>`
- Scatter Core Chart: `<gvisualization:scatterCoreChart/>`
- Stepped Area Core Chart: `<gvisualization:steppedAreaCoreChart/>`
- Candlestick Core Chart: `<gvisualization:candlestickCoreChart/>`
- Combo Core Chart: `<gvisualization:comboCoreChart/>`
- Gauge: `<gvisualization:gauge/>`
- Table: `<gvisualization:table/>`
- Map: `<gvisualization:map/>`
- Annotated Time Line: `<gvisualization:annotatedTimeLine/>`
- Organizational Chart: `<gvisualization:orgChart/>`
- Intensity Map: `<gvisualization:intensityMap/>`
- Geo Map: `<gvisualization:geoMap/>`
- Geo Chart: `<gvisualization:geoChart/>`
- Motion Chart: `<gvisualization:motionChart/>`
- Tree Map: `<gvisualization:treeMap/>`
- Timeline: `<gvisualization:timeLine/>`
- Calendar Chart: `<gvisualization:calendarChart/>`

4.1.2. Deprecated Charts

- Pie Chart: `<gvisualization:pieChart/>`
- Bar Chart: `<gvisualization:barChart/>`
- Column Chart: `<gvisualization:columnChart/>`
- Area Chart: `<gvisualization:areaChart/>`

- Line Chart: `<gvisualization:lineChart/>`
- Scatter Chart: `<gvisualization:scatterChart/>`