Google™ 09 I O

Using the Visualization API with GWT and Other Advanced Topics

Itai Raz

May 27, 2009

Agenda

- Visualization API & GWT
- More Advanced Topics
    - Latency
    - Security / Privacy
    - Data View
- Q&A

The Google Visualization API

- A tool set to enable visualization of structured data on web pages
- javascript api
- A gallery of many visualizations

The Javascript API Concept

- Many visualizations, various implementation
- Same way of loading to the page
- Same API:
  - `new chart(div);`
  - `chart.draw(dataTable, options);`
- Same event mechanism
- Option to get the data from external data sources
- 3rd party charts can be wrapped easily

So All You Have To Do....

- Load the visualization(s)
- Create a data table
- Draw it !

GWT Visualization API

Using a Visualization

```
public class SimpleVizDemo implements EntryPoint {

  public void onModuleLoad() {
    Runnable onLoadCallback = new Runnable() {
      public void run() {
        DataTable dataTable = createTable();
        PieChart.Options options = createPieOptions();
        PieChart pie = new PieChart(dataTable, options);
        RootPanel.get().add(pie);
      }
    };

    VisualizationUtils.loadVisualizationApi(onLoadCallback,
        PieChart.PACKAGE);
  }

...
}
```

Creating a Data Table

```
public class SimpleVizDemo implements EntryPoint {

...

  private DataTable createTable() {
    DataTable data = DataTable.create();

    data.addColumn(ColumnType.STRING, "Task");
    data.addColumn(ColumnType.NUMBER, "Hours per Day");
    data.addRows(2);
    data.setValue(0, 0, "Work");
    data.setValue(0, 1, 14);
    data.setValue(1, 0, "Sleep");
    data.setValue(1, 1, 10);

    return data;
  }

}
```

Setting Configuration Options

```
public class SimpleVizDemo implements EntryPoint {

...

  private PieChart.Options createPieOptions() {
    PieChart.Options options = PieChart.Options.create();

    options.setWidth(400);
    options.setHeight(240);

    return options;
  }

}
```

Sending a Query to a Data Source

```
String dataUrl = "http://my.datasource.com";
Query query = Query.create(dataUrl);

query.send(new Callback() {
  @Override
  public void onResponse(QueryResponse response) {
    if (response.isError()) {
      Window.alert(response.getMessage());
    } else {
      DataTable data = response.getDataTable();
      PieChart pie = new PieChart(data, null);
    }
  }
});
```

Handling Events

```
pie = new PieChart(dataTable, options);
pie.addSelectHandler(createSelectHandler(pie));

private SelectHandler createSelectHandler(
    final PieChart chart) {
  return new SelectHandler() {
    @Override
    public void onSelect(SelectEvent event) {
      // Apply your logic here
      bar.setSelections(chart.getSelections());
    }
  };
}
```

Unwrapped Options

```
public class SimpleVizDemo implements EntryPoint {

...

  private PieChart.Options createPieOptions() {
    PieChart.Options options = PieChart.Options.create();

    options.setWidth(400);
    options.setHeight(240);

    options.setOption("is3D", true); // options.set3D(true);

    return options;
  }
}
```

GWT - Wrapping a Visualization

It is possible to wrap a javascript visualization that is not wrapped by the library

Hint: The library is Open Source, so there are no secrets

GWT - Creating a Visualization

It is even possible to write a new visualization from scratch using GWT!

Other advanced topics

Improving latency

Latency Factors

Before the chart is displayed, three steps happen:

- The chart (and all resources) is loaded
- A data table is created and populated
- The chart is drawn

It is possible to speed up the first two steps

Loading and Auto-loading

The javascript library is loaded from the main loader servers of Google at http://www.google.com/jsapi

```
<script src="http://www.google.com/jsapi"></script>

google.load('visualization',
            '1',
            {'packages': ['piechart']});

google.setOnLoadCallback(drawChart);
```

Autoloading (*new!*)

Saves one HTTP request (although somewhat less readable)

```
<script src="http://www.google.com/jsapi?autoload=
  {'modules':
    [{'name':'visualization',
     'version':'1.0',
     'packages':['annotatedtimeline']}]}"></script>

google.setOnLoadCallback(drawChart);
```

JSON vs JS Data Table Construction

Two ways to populate a data table

- Javascript is much more readable

```
var data =
  new google.visualization.DataTable();

data.addColumn(...);
...
data.addRows(...);
data.setCell(...);
data.setCell(...);
...
```

JSON vs JS Data Table Construction

- JSON significantly improves latency (x2)

```
var json = {
 cols: [{id: 'c1', type: 'string'},
        {id: 'c2', type: 'number'}],
 rows: [{c:[v: 'Joe', v: 100]}]};

var data =
  new google.visualization.DataTable(json);
```

Security / Privacy

Script Injection / Restricted Mode

How to send a query to a data source?

- Script Injection - The only method to date
- Generates a dynamic javascript tag
- Advantage
    - Enables cross domain requests (gadgets)
- Disadvantage
    - Enables cross domain requests (xsrf)

Script Injection / Restricted Mode

How to send a query to a data source?

- Restricted Mode - New addition to the protocol
- Using xhr (AJAX) to send HTTP GET request
- Adding Custom Header
- Should be used together with cookie based authentication
- Data accessible to authenticated users, on your site only
- Implemented in the new Java data source library

Script Injection / Restricted Mode

New optional parameter to Query.send()

```
function sendQuery() {
  var query = new google.visualization.Query(url);

  query.send(func, {sendMethod: 'xhr'});
  // OR
  query.send(func, {sendMethod: 'script_injection'});
}
```

Google Secure Data Connector (SDC)

- SDC enables your Google services to access your private network data
- You have to install an agent
- The Google Visualization API lets you use the standard gadgets over Google Sites, to access that data without any extra configuration
- Add to you data source url `&tqrt=makerequest`

Security / Privacy - Data Policy

Some visualizations may send the data away from the browser to a server in order to generate the chart

- Image based charts
- 3rd party charts (some of them)

Other visualizations do not send any data from the browser

- "Interactive charts"
- Flash based charts (Annotated Time Line, Motion Chart)

Data policy of each visualization is specified in the charts gallery

DataView

DataView

- Same 'interface' as of DataTable
- Used by visualizations in the same way

```
function drawVisualization() {
  var pie = new google.visualization.PieChart(div);

  var dataView =
      new google.visualization.DataView(dataTable);

  dataView.setColumns([0, 2]);

  pieChart.draw(dataView, options);
}
```

DataView - Row Filtering

- It is possible to filter the rows to be used
- By default - all rows are selected

```
function drawVisualization() {
  var dataView =
      new google.visualization.DataView(dataTable);
  dataView.setColumns([0, 2]);

  var rows =
   dataTable.getFilteredRows([{column: 2, value: 42}]);

  dataView.setRows(rows);

  pieChart.draw(dataView, options);
}
```

Data View

Last quick example....

Tomorrow - How to Implement a Data Source

Using the new open source java library, implementing a data source is as simple as creating a data table on your server

Thanks!

Q & A

Google™ 09 I O

Appendix