GViz: A Web Based Client Side Data Visualization Framework

M.W. Edirisooriya, B.P.P. Fernando, T.A.M.P. Fernando, W.A.W.S.K. Wickramasinghe and S.M Weerawarana
Department of Computer Science and Engineering,
University of Moratuwa,
Moratuwa, Sri Lanka.

Abstract— **Data visualization is widely used in modern day web applications. Although there are popular solutions like Google Visualization API and YUI, they all are dependent on the server side technology. This paper describes a generic framework which runs entirely on the client side of a web application and enables easy integration of different types of visualization components like maps and charts from different visualization providers into a single web based dashboard [1] supporting different types of data providers like XML and JSON [2]. Data is described to the framework using an open standard named Data Set Publishing Language (DSPL) [3]. The framework implements an Event Driven Architecture (EDA) to facilitate the numerous asynchronous workflows happening inside.**

Keywords—**data visualization; client side scripting; data description; JavaScript framework; Data Set Publishing Language (DSPL)**

I. INTRODUCTION

Data visualization is one of the main requirements for professionals such as economists, data analysts, scientists, researchers and students. They need to visualize different types of data (e.g.: XML and CSV files) in different types of visualization techniques (e.g.: Bar charts and maps). There are both open and proprietary software available for those visualization purposes. Most of such software are desktop based, platform dependent and specialized for particular applications. For example Microsoft SharePoint [4] is designed for business data visualization, MATLAB visualizations are designed for mathematical data visualization. Geographic Information System (GIS) [5] are designed for visualization of geographic data. Therefore a user should have a specialized knowledge to get use from the above mentioned complex software. As they are available as parts of software, it is difficult to use their visualization components (e.g.: Pie charts and maps) with other software. For example if the user use Microsoft SharePoint then he/she must use Microsoft Windows operating system with .net framework as a dependency. Also, the integrator should learn the API provided by the software. It becomes much difficult if the integrator wants to bring both SharePoint and MATLAB visualization components into a single interface. As the alternatives to the above mentioned problems the visualization technology has moved from the desktop to the web with the development of technologies such as Flash, HTML5 and AJAX. As these technologies are common to most of the currently available web browsers they can be used as platform independent visualization techniques. Google Visualization API [6] and Google Dashboard [7] are examples for such web based solutions for data visualizations. But still the technology is completely dependent on the server side support provided by the server (In the above examples, Google server). That requirement raises a lot of problems.

The user cannot use the software without an Internet connection.

1. Internet bandwidth is a main performance issue.
2. Reliability of the software is dependent on the service provided by the server side company.
3. Privacy and confidentiality are severely compromised on the data to be visualized.
4. User cannot improve and customize the visualization software according to his/her requirements.

We have identified those problems [8] and came up with the solution of "Client Side Data Visualization Framework" which is novel to the visualization technology. The main advantage of this framework is its capability to reuse existing visualization components used in other web based technologies such as visualization components used in Google Visualization API, Highcharts [9] and Ovi Maps [10]. Also, its data source can be given in a variety of ways in different formats such as CSV files, XML files and JSON files. The other advantage is that the user can map these data sources to

appropriate visualization components using a configuration file in Dataset Publishing Language (DSPL). Also the user can define his own data source and he/she can reuse existing web based visualization components by adding them to the framework by writing a custom adapter.

II. DATASET TECHNOLOGY

According to the aim of our project, we need to visualize data obtained from different types of data sources in different formats. Types of data sources are as follows.

1. Files - Provided from server by AJAX requests or directly from the local machine
2. Online document - Using the URL to the document and access privileges
3. Web service - Can be enabled using the existing XML string data provider
4. Database - Should be given from the server or can use the local storage defined in HTML5
5. Manually entered data - Application should enable an interface for the user to enter data

Data formats that are already implemented in the framework are as follows. They are defined in our framework as Data Providers.

A. Tables

The most generic data type is the Table type. It is easy to directly convert these types of data into other data types. This format can be created by manually entered data and directly accessed from an online document such as a Google Spreadsheet. That data provider is known as "gdoc" data provider in our framework.

B. XML format

DSPL and source data are retrieved in XML format. There are two data providers for both XML files and XML strings. XML string data provider can be extended as a web service data provider that can interact with web services.

C. JSON format

JSON format is the inherent object notation used in JavaScript. As all the input data is converted into JSON format it can easily provide source data in JSON format as well. For this format also, there are two data providers available for both JSON files and JSON string formats.

D. CSV format

Usually large amounts of data are stored in files in CSV format. CSV source data files can be given to the framework using the CSV data provider.

The data given in the above formats should be according to the data definition specified in the DSPL file. All these data providers convert its input data into the JSON format inside the framework and store them in the GViz data table which is a data structure defined in our framework. DSPL specification is used for this conversion. After this conversion it is very easy to provide appropriate slices to the required visualization components as accessing JSON format is simple in JavaScript.

III. VISUALIZATION COMPONENTS

The main visualization software components used in the framework is known as visualization components. As mentioned earlier a visualization component consists of,

1. A web based UI component developed in the web technology that may be already available in some other software
2. A custom JavaScript adapter mapping the functionality and data source of the UI component to the framework

There are some reused visualization components already available in the framework such as from the components from Google Visualization API, Highcharts, Bing Maps [11], Ovi Maps and Google Maps etc. The user too can add their own visualization components or reuse existing visualization components and write an adapter in JavaScript to make it compatible to our framework.

IV. DATASET PUBLISHING LANGUAGE (DSPL)

This is an open standard introduced by Google on February 2010. DSPL is an XML configuration standard defining data in the data source and mapping them to visualization components by assigning each slice of data to each visualization component in Google Public Data Explorer [12]. As our requirement is broader than the goals of Google Public Data Explorer we extended the DSPL framework to be able to assign some additional features like mapping to the exact div of the visualization component in an HTML file. We also made it possible to use exact pure DSPL defined by Google for the advanced users and made it possible to handle the extra complexity using the JavaScript code.

As a supporting service to the visualization framework we developed a server side code generator and a DSPL generator for the use of ordinary users. Using that framework they can use an existing data source (data provider) and an existing visualization component to fulfill their visualization requirements.

Code generator generates the JavaScript code and the DSPL generator generates the mapping DSPL code between the data provider and the visualization component.

V. Architecture And Flow

A. Overall Architecture

The framework runs on client side of the web browser. Figure 1 refers to the overall architecture of the framework. It has a data structure used to keep the DSPL Description globally that is needed for all the other events. In the same object it has the DSPL access interface that can access the content of the DSPL description.

Then there is a data table implemented in the object GViz Data Table that keeps the data for all the visualization components. Data given to each visualization component is a single slice from the table of GViz Data Table. This encapsulates the confidential data for each visualization component.

Data for the GViz Data Table are filled with the data providers of different types. For example, XML File data provider supplies data from an external XML file in the given location. All of these data providers work independently and asynchronously [13] from each other. As the DSPL description is an XML file it is loaded using the XML File data provider.

Each visualization component is provided with data in a single slice as described in the DSPL description. As GViz Data Table provides a single API for the visualization component, custom written visualization adapters are used to communicate between visualization components and that API.

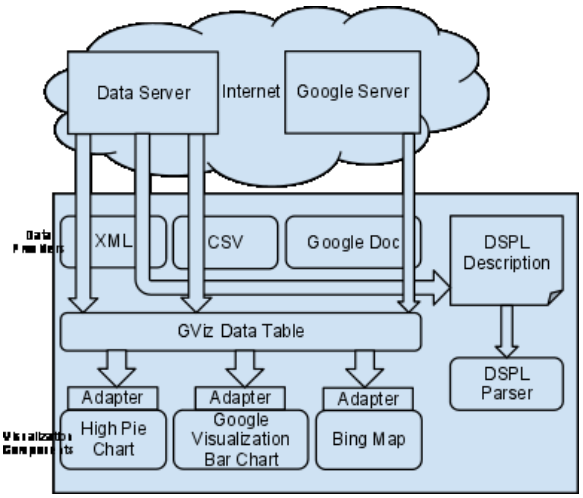

Figure 1 - Overall Architecture

B. Event Mechanism

According to the Figure 2 as the framework uses a lot of asynchronous calls using AJAX and secondary storage transactions the framework is designed using Event Driven Architecture (EDA).

When the framework loads, first it loads the DSPL file from the given location (DSPL configuration loaded event). According to the specification of the DSPL file, then it loads the external dependencies that are required mainly for the custom visualization components. This event is known as Dependency configuration loaded event. Then the framework loads required data sources from appropriate data providers according the DSPL specification. All loaded data are first converted to a standard JSON format and then bind to the GVIz data table. Headers of each data are assigned in the GVIz data table according to the DSPL specification. Finally all the loaded data are used as the data sources of the visualization components and the visualization components are loaded from the given location asynchronously. After this event, visualization component loaded event is fired. This is the event model.

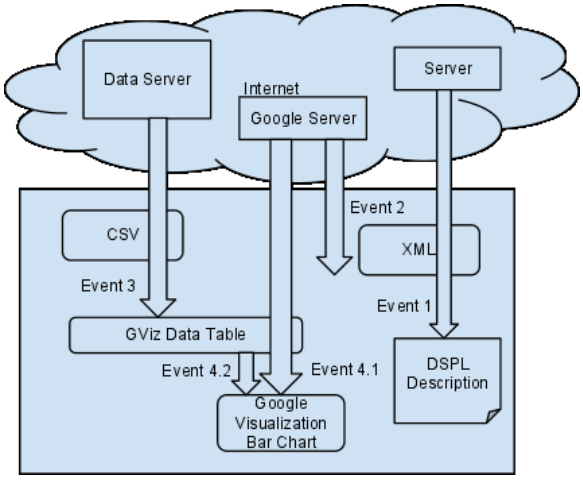

Figure 2 - Event Sequence

Event 1 - DSPL configuration loaded event
Event 2 - Dependency configuration loaded event
Event 3 - Data loaded event
Event 4.1 and 4.2 - Visualization component loaded event.

All events are fired sequentially in the given order. Each event is handled by the framework and

continues the framework until the next event. These events are asynchronous so that the framework waits for the event without any execution. Event 1 handles only one call as there is only one DSPL file but in event 2 it can be many dependencies as the framework may need many dependencies for further execution. Event 3 may be one or more calls as there may be one or more types of data to be loaded. As there can be many visualization components used in the framework Event 4 can be many events.

C. *Visualization Component Interaction*

Although each visualization component is built isolated from each other using adapters, there is a global mechanism for visualization components to interact with each other at run time. This mechanism is achieved using a global event mechanism. For example, as shown in Figure 3, assume a location of a map visualization component is selected. That on-click event is wrapped with a global event with the relevant longitude and latitude and fired globally. If a table visualization component has previously subscribed for that event, it can catch the event and select the related row asynchronously.

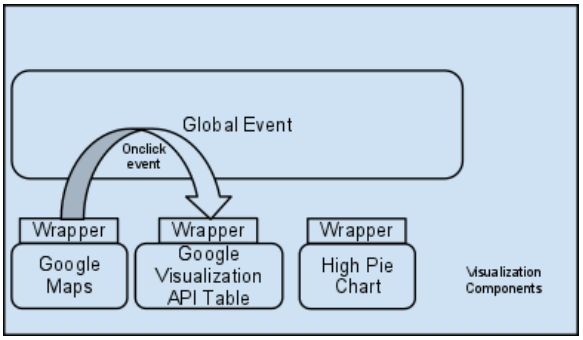

Figure 3 - Visualization Component Communication

VI. CONCLUSION

The problem of generic data visualization requirement can be successfully fulfilled with the solution of a client side data visualization framework. There are external on the shelf visualization components that can be integrated to the framework using a custom adapter. All the visualization components can be filled with data from a single type of data structure independent from the data source. Multiple visualization components are loaded asynchronously because of the application of event driven architecture.

A common data source definition is achieved with the open standard, DSPL. It enables the framework to provide data as slices from the data table improving the isolation of data for the requirements of security. And also DSPL acts as a single configuration for all the data sources and their mapping to each visualization component.

Interaction among visualization components is a major requirement when it comes to complex dashboards. This problem is handled in this framework with the use of global event system accessible for all visualization components. Requirements of access control and generality can be achieved with the wrapping of global events with a restricted event object.

ACKNOWLEDGEMENT

At this very important achievement of our academic careers we would like to remember all the members in our families for their undying support in every endeavor in our lives.

Next we would like to acknowledge our project supervisor, Dr. Shahani Weerawarana for the support from beginning, providing the initial project idea and guiding us to make the geo-data visualization framework a reality.

Then our special thanks go to Level 4 Project Coordinator, Dr. Shantha Fernando for commitment from scheduling all the project related evaluations in the correct time to providing correct feedback after careful review of the project.

Then our gratitude goes to all the lecturers and Staff members of the Computer Science and Engineering Department for their dedicated support in educating us to become more knowledgeable human beings.

Last but not least, we would like to remember all our batch mates in the department for the unforgettable time we had as CSE family members.

REFERANCES

[1] F. B. Viégas, M. Wattenberg, F. V. Ham, J. Kriss, M. McKeon, “Many Eyes: A Site for Visualization at Internet Scale”

[2] “Introducing JSON”. [Online]. Available: http://www.json.org/. [Accessed: 10-Oct-2011].

[3] “DSPL: Dataset Publishing Language”. [Online]. Available: http://code.google.com/apis/publicdata/. [Accessed: 10-Oct-2011].

[4] “Product Information”. [Online]. Available: http://sharepoint.microsoft.com/en-us/product/Pages/Features.aspx. [Accessed: 10-Oct-2011].

[5] “What is GIS? | GPS Systems,” 25-Aug-2011. [Online]. Available: http://gpssystems.net/what-is-gis/. [Accessed: 25-Aug-2011].

[6] “Introducing the Google Visualization API” 19-Mar-2008. [Online]. Available: http://googlecode.blogspot.com/2008/03/introducing-google-visualization-api.html. [Accessed: 10-Oct-2011].

[7] "Transparency, choice and control — now complete with a Dashboard!" 05-Nov-2009. [Online]. Available: http://googleblog.blogspot.com/2009/11/transparency-choice-and-control-now.html. [Accessed: 10-Oct-2011].

[8] C. Weisgerber and S. H. Butler. "Visualizing the Future of Interaction Studies: Data Visualization Applications as a Research, Pedagogical, and Presentational Tool for Interaction Scholars" The Electronic Journal of Communication, vol. 19, pp. 1-2, Nov. 2009.

[9] "Highcharts JS". [Online]. Available: http://www.highcharts.com/. [Accessed: 10-Oct-2011].

[10] "Ovi by Nokia". [Online]. Available: http://www.ovi.com/. [Accessed: 10-Oct-2011].

[11] "Bing Maps". [Online]. Available: http://www.bing.com/maps/. [Accessed: 10-Oct-2011].

[12] "Google Public Data Explorer". [Online]. Available: http://www.google.com/publicdata/home. [Accessed: 10-Oct-2011].

[13] J. Heer, F. B. Viégas and M. Wattenberg. "Voyagers and Voyeurs: Supporting Asynchronous Collaborative Information Visualization".