

Client-Server Communication in SVG Web Applications

Jan-Klaas Kollhof

University of Rostock

Contents

Keywords: SVG, Web Applications, Client-Server,

Abstract

SVG¹ is a rich language for describing 2D graphics and animations. It has evolved to become a framework, allowing the developer to create web based client applications. This paper will have a look at SVG's networking features, necessary to build web applications. It is intended to be a companion paper for the "SVG as a tool for Presentation and Communication in Business and Education" paper², presented at the SVGOpen 2004 in Tokyo.

1 Introduction

SVG provides a feature rich language to describe two dimensional vector graphics. With its interactivity, platform independence and its scripting features, it is well suitable to build web applications. This paper is a companion paper of the "SVG as a tool for Presentation and Communication in Business and Education" paper, presented at the SVGOpen 2004 in Tokyo. Its intend is to give some background information on client server communication in SVG.

2 SVG's networking features

In order to build web applications the technology used must provide ways for communication between the client and the server part of the system. When SVG was first introduced the authors of the specification did not include any networking functionality. There was no way to transmit or retrieve data to and respectively from the server besides using links in the SVG document, which when clicked would result in the viewer requesting a new document from the server transmitting data through the URL.

¹ more information on Scalable Vector Graphics (SVG) can be found at <http://www.w3.org/Graphics/SVG/>

² See <http://www.svgopen.org/2004/papers/ToolForPresentationAndCommunication/>

2.1 getURL and postURL

Adobe first introduced an extension to SVG, which they built into their plugin component. It consisted of mainly two functions exposed to the scripting environment: `getURL` and `postURL`. Using these functions in ECMAScript one was now able to retrieve data from a server using a simple HTTP GET or post data to the server using an HTTP POST. The connections were asynchronous, making sure the script engine would not lock up during the request. Using a callback object the data returned by the server could then be processed in the SVG application.

2.2 XMLHttpRequest

The non-standard extensions `getURL` and `postURL` have been widely accepted. Though convenient, they lack certain features. The authors of the SVG1.2 specification have addressed this problem and added DOM interfaces for URL-requests³. A `XMLHttpRequest` object is a DOM `EventTarget`, allowing an application to add `EventListeners` that listen for `XMLHttpRequestResponse` events. The object exposes methods to set and read headers, to be sent to the server and respectively received from the server. There are methods to initialize a request, one to submit it and a method to abort the request.

2.3 Sockets

URL requests are a simple way of communicating with a server, though they only allow one to use HTTP. To allow the SVG application to use different protocols, the SVG1.2 specification have introduced raw sockets. The socket object again is an `EventTarget` and listeners registered on that object may listen for `ConnectionData` events. In addition, the object exposes a number of methods to open and close a connection as well as a method to send data.

3 Protocols

There are many protocols available for client server communication. For web applications written in SVG the most common one besides custom encoding protocols is URL-encoded data. It is the same as used when HTML forms are submitted. There are a number of implementations and tools already available, which

³ SVG 1.2 at <http://www.w3.org/TR/SVG12/>

makes it cost effective and simple to use for development. More convenient protocols allow the mapping of local methods to remote methods, making remote method calls behave in a way as if local methods were invoked. Type information of objects passed to the remote service and objects returned are preserved. At the present time, there are implementations of two RPC protocols to be used in SVG.

3.1 XML-RPC

There is the W3's SOAP, the Simple Object Access Protocol. Despite it's name it is rather complex. XML-RPC⁴ truly is a simple remote procedure call protocol. It has six simple data types: boolean, int, double, string, ISODate and base64; and two compound data types: struct and array. Compound data objects may contain objects of any type simple or compound. Calling a method of an XML-RPC service is accomplished by a ServiceProxy object which sends requests to a server hosting a service. A request is XML data containing the name of the method to be invoked on the service and the parameters to be passed to that method. The data is sent to the server using a HTTP POST request. The server processes the data received by unmarshalling the XML to suitable data objects and invoking the desired method passing in the parameters. Any result returned by the method or an exception raised is transformed into XML and returned to the client. The client unmarshals the data and returns the result to the calling object.

XML-RPC has been implemented in various languages including ECMAScript which makes it suitable to use it from within the scripting environment of an SVG document⁵. Examples for applications using XML-RPC or providing services are Zope, blogging services, O'Reilly Meerkat, NewsIsFree.com, web applications at the Austrian Air Traffic Control, web front end for accessing NMR spectrometers in Research and Development labs. to name a few.

3.2 JSON-RPC

XML-RPC and posting URL-encoded data use HTTP for transportation. Due to the nature of HTTP, it is impossible for a server to notify a client of an event. A client application would need to constantly poll data from the server to find out if an event has occurred or not, thus using up resources and unnecessary network traffic.

⁴ XML-RPC can be found at <http://www.xml-rpc.com>

⁵ XML-RPC implementation can be found at <http://xmlrpc.kollhof.net>

JSON-RPC⁶ is a protocol especially designed to overcome this problem. It uses JSON, the JavaScript Object Notation, to encode data and socket stream connections for transportation. It is also possible to use JSON-RPC over HTTP, which is similar to XML-RPC with the difference of the data encoding used. When using JSON-RPC over socket streams a connection has to be established between two peers. Once a connection is established, both can communicate with each other by invoking methods provided by services connected to the peers. Each peer may invoke methods of the remote service at any time. Such an invocation is processed by creating a request object which carries an ID to identify it, the name of the method to be invoked and the parameters to be passed to the method. The request is encoded using JSON and sent over the stream to the peer. The peer decodes the data and calls the method specified passing in the parameters provided. The result of the call or any exception raised is stored in a response object, which has the same ID as the request. The response is encoded using JSON and sent back to the calling peer. The response will be decoded and the result is passed to the calling object. A special form of a request having no ID is a notification. A notification may not be replied to with a response. JSON-RPC is a very simple, yet powerful remote procedure call protocol. An example in python will show how the protocol works and how it is used:

```
class LocalService:
    def printMsg(self, msg):
        print msg

localService = LocalService()
remoteService = jsonrpc.ServiceProxy("jsonrpc://json-rpc.org:1234",
    localService)
remoteService.callMyPrintMsg("Hello JSON-RPC")
```

In the example a localService object is created for the remote peer to call methods of. The script then creates a remoteService object which will invoke the method of a remote service using the JSON-RPC protocol. It then calls the remote method, callMyPrintMsg, passing in a string. The data transmitted between the peers looks as follows. a -> is used to indicate data sent from the script to the remote service and a <- is used to indicate data sent from the remote service to the script:

⁶ For more information see <http://json-rpc.org>

```
-> {'id':'1', 'method':'callMyPrintMsg', 'params':["Hello JSON-RPC"]}
<- {'id':'1', 'method':'printMsg', 'params':["Hello JSON-RPC"]}
-> {'id':'1', 'result':null, 'error':null}
<- {'id':'1', 'result':null, 'error':null}
```

One can see the protocol is very verbose, yet it has less overhead than SOAP or XML-RPC. And similar to other RPC protocols it is bidirectional, allowing a peer to invoke methods of its connected peer. Presently, there are implementations available in ECMAScript, Java and python.

4 Use Cases

The paper "SVG as a tool for Presentation and Communication in Business and Education" describes an application which makes heavy use of JSON-RPC. It is a tool which synchronizes presentation slides on multiple clients, allowing the user to make annotations which are stored on the server and providing communication services like text chat for the user. It can be used as a discussion tool or a presentation application. Because the client components of the system are pure SVG, it is very portable only requiring an SVG viewer or plug in to be installed on the users machine.

Life mapping solutions are another example for SVG and client server communication. Map data is loaded from the server on demand. Positioning data of objects on the map can be updated in realtime. For example: life flight navigation maps showing all aircrafts and their flight procedures in a selected region. The map is continuously updated using data from onboard equipment and ground control information which has been received through e.g. a socket connection to a data server.

Life charting visualizations may use SVG's rich vector features to display charts. The data for the charts is acquired from a server using the SVG networking features.

For web based training and e-learning it is important to use web standards. With SVG's rich 2D graphics, its ability for animation, displaying video and playing sound, its interactivity features, scripting and its networking interfaces it is possible to build platform independent application systems that are able to communicate with backend systems to interface with databases, communication tools and other server based application logic.

5 Conclusion

This paper showed what networking features SVG provides and how they can be used for client server communication. There are implementations of simple RPC protocols to ease development and decrease engineering costs. In addition, using SVG editors and conversion tools SVG provides a powerful tool to build web based application systems.