Snap! Connectivity Strategy

Jens Mönig
Brian Harvey

August 16, 2012

Summary

Snap! 4.0 provides connectivity to remote resources through HTML5 (XHR2) XMLHttpRequests, both in the Snap! application itself and in user authored projects. This allows Snap! to be extended by modules hosting access to databases, robots, sensors, cameras etc., and to even develop interfaces to such modules in the Snap! language itself. Web servers hosting resources to be interfaced by Snap! must ensure that these are shared conforming to the W3C CORS (Cross-origin resource sharing) working draft.

Objective

Snap! is an HTML5 application running in web browsers, and as such it is subject to various sandboxing restrictions, e.g. it cannot access the local file system or web resources hosted on another domain.

This has lead other developers wishing to extend Snap! with access to specialized hardware such as robots and sensors to distribute their own forks of Snap! and to serve them through locally installed web servers. As a result, user projects created in such specialized forks of Snap! are incompatible and cannot be run in the main Snap! Also, developers of such forks are forced to constantly monitor the development of the main Snap! trunk and to manually adjust their branches accordingly, if they wish to benefit from new features, error corrections or other improvements.

Another important objective is to connect the Snap! application to a powerful backend which will enable saving and retrieving projects online („Snap Cloud“), and to create a social network website through which users can collaborate in developing SNAP projects, share them among user groups (e.g. school classes) or publish them to the world. Rather them having to embed all of Snap! into the backend framework both should be implemented as individual SaaS entities and hosted on their own domains, allowing for greater flexibility and maintainability.

Snap! is a copy-lefted open source project. By offering a uniform method for Snap! to interface with modules hosted on different domains, such extensions do not have to be open source themselves but can also be implemented as proprietary software, encouraging commercial vendors to invest in developing Snap! extensions.

Cross-Origin Resource Sharing

Enabling Snap! (or any other JavaScript client) to access resources hosted on another server is easy. All it requires is adding another HTTP response header to the server hosting the remote resource.

For example:

```
Access-Control-Allow-Origin: *
```

permits the resources to be accessed by clients hosted on any site.

Servers wishing to be more specific about the client‘s origin can limit access to scripts hosted on a space-delimited list of servers, e.g.:

```
Access-Control-Allow-Origin: http//snap.berkeley.edu http://chirp.scratchr.org
```

Browser Support

Both XMLHttpRequest and CORS are supported by the current versions of all major web browsers. We have conducted tests with a local (downloaded) version of Snap! and one served from http://snap.berkeley.edu/run using a CORS web server written in Python and were able to confirm that cross-origin resouces could be accessed by XMLHttpRequests in the Chrome, Firefox, Safari and Opera web browsers.

Hardware Extensions

Hardware such as the Kinect camera, the Finch robot, medical lab equipment etc. can provide their own „middleware“ clients to be downloaded and installed by Snap! users on their local computers. We recommend writing such „middleware“ in scripting languages such as Python, which are supported by most platforms, obliterating the need to offer OS-specific downloads. Middleware clients are both (local) web servers and also handle communication with their hardware. They implement a URL-based polling protocol for Snap!, and provide text-based parsable responses (e.g. XML).

Web Links

http://en.wikipedia.org/wiki/Cross-origin_resource_sharing
http://www.w3.org/TR/cors/
http://enable-cors.org/