*F*ire*F*ly: Lightweight AJAX System for F2F-CL

Rosario De Chiara, Marco Volpe

ISISLab - Dipartimento di Informatica ed Applicazioni
Università di Salerno,
84084, Fisciano (SA) – Italy
E-mail: dechiara@dia.unisa.it, volpemar@gmail.com

Abstract. In this paper is illustrated the development of *F*ire*F*ly, a system for the *Face to Face Collaborative Learning* (F2F-CL). *F*ire*F*ly is written using AJAX, a set of technologies that allows to create rich web-based application. The challenge is to develop a system able to be executed on usual desktop PC, requesting the lesser possible installation effort and achieving enough expandability to allow further extensions. The system is currently under development, but the prototypes and the early tests confirms that the use of AJAX technology for this kind of application is suitable and deserves further investigations.

1 Introduction

In this paper we report the results of our experiences in implementing a system for F2F-CL, *F*ire*F*ly, exploiting the set of technologies known under the name of AJAX. In our intentions the system must provide the following features:

- Easiness both in the installation and management: virtually *zero* installation effort. This is particularly critical considering that installations are usually made on tens of personal computers.
- Low-end hardware: this can be an issue in various scenarios, for instance schools.
- No Internet connection required: Internet connection is not available everywhere and even where it is available it can be subject to heavy controls and limitations.
- Need for a cross platform solution: platform independence is appreciated from the point of view of the developer, the same application for every machine, and from the point of view of the user that has not to face the problem of different *User Interfaces* (UIs).

The intersection of all these limitations excludes totally or partially some traditional solutions, for various reasons:

E. Tomadaki and P. Scott (Eds.): Innovative Approaches for Learning and Knowledge Sharing,
EC-TEL 2006 Workshops Proceedings, ISSN 1613-0073, p. 59-66, 2006.

- Java applet downloaded from a server located somewhere in the world: this solution would allow to avoid to face legacy hardware (at least for the server) allowing to provide high level services through traditional Java applet software. This kind of solution rely on a trustworthy Internet connection that is not always available. On the other hand Java applications are, of course, cross platform.
- Local Java applications: this solution is suitable for installations on machine not connected to the Internet, but can meet the limitations of computers on which the software cannot be installed. The "install nothing" policy is often enforced by system administrators as a radical solution to viruses and malware.
- Native language solution: a native language solution is not cross-platform by definition. As a pro it can deeply exploit the hardware.

Starting from all these considerations we have pondered about the use of AJAX technology would present some interesting points on its favor.

2 AJAX

The term AJAX is the contracted form for the expression *Asynchronous Javascript And XML* [13]. AJAX is a set of technologies at the hearth of which there is the capability of modern browsers (Mozilla Firefox, Microsoft Internet Explorer, Apple Safari, Opera) of managing an API (*Application Program Interface*) called `XMLHttpRequest`. This API, available within browser through Javascript, allows to transfer data to and from a web server using HTTP. This data transfer is carried out over an independent connection channel and the moved data are formatted in XML.

`XMLHttpRequest` is particularly important because it allows *asynchronous transfers* of data between the client and the server, and this permits to break the constrain of using the traditional form submission mechanism used in HTML [6]. Using `XMLHttpRequest` is just one of the elements that made up the AJAX technology, after the data are moved asynchronously between client and server the next step is to update the user interface in order to reflect the results of this data exchange. The user interface is managed through an HTML web page that is in charge for displaying data and gathering user inputs. Being the data updated asynchronously the user interface must be update in the same manner without a page reload. To obtain this, two well known ways of designing web pages are used: XHTML and CSS, in order to define the styles of the various components (text, labels, buttons etc. . .), and DOM(*Document Object Model*) to address the components of the page that are intended to be modified.

What is really new in AJAX is not the set of techniques but the way this techniques are used to meet a goal that is to use the browser and the network as *a platform for implementing interactive web applications*. The combined effort of `XMLHttpRequest` for exchanging data and the dynamic look and feel provided to the web pages by the use of XHTML and DOM enable the developer to create applications like GMail [5], Writely [10] and YouOS [12]: GMail is a web mail application, Writely is a cooperatively usable word processing and YouOS tries to mimic the basic behaviors of an operating system (actually a window manager).

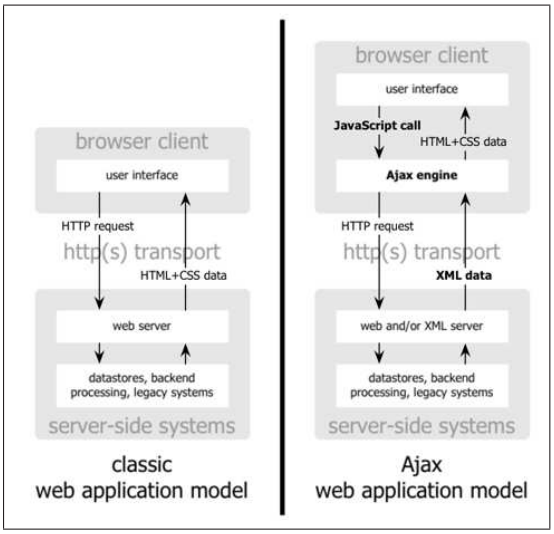

Fig. 1. Comparison between the classic web application and the AJAX web application model. (Image from [13])

In Figure 1 is compared the classic web application model and the AJAX model, on the left is shown the traditional form-driven application, while on the right there is the AJAX model in which the AJAX engine presence within the browser is emphasized. The data exchange between the client and the server is XML and this clearly request a server able to parse and create well-formed XML to send it back to the client.

3 System description

The system is implemented through a suitable configured web server that will provide the application to the clients through an HTML page. A client

for *F*ire*F*ly is any modern web browser with not particular settings at all. To let an ordinary PC act as a server for some tens of clients we have focused our attention on finding a computational light web server that would not request too much computational effort.

3.1 Functionalities

The system architecture is quite simple, the software is installed together with the web server. The server part is written in PHP while the client part is Javascript, being it executed in a browser. To boot the system it is enough to execute the web server that simply will be waiting for connections. The participants can enter the system just pointing their browser to the server IP address. An authentication screen will be presented to log into the system.

The system currently provides two tools for the collaboration. One tool is a traditional unstructured chat in which contribution from users are just appended. The other tool is a threaded chat in which the contributions can be structured in a hierarchical manner.

Another task carried out by clients is the gathering of all input, this is performed by Javascript functions in execution within the browser.

Worth nothing is the fact that the system does not use any database engine, everything is stored in XML files and in order to avoid wasteful parsing of huge XML files containing more days of interactions, files are timestamped and rotated everyday in order to keep their size reasonable. These XML files could be used as input for trace analysis softwares a limitation of these traces is that they are coarse grained because of the architecture of *F*ire*F*ly.

3.2 Lightweight web servers

Clients interactions will be managed from the web server through CGI (*Common Gateway Interface*) scripts written in PHP. The ability of running CGI scripts is the sole feature a web server has to provide to host AJAX applications. We have compared three solutions, choosing among light web servers: Sambar Server [8], lighttpd [7] and ghttpd [4].

Sambar Server Sambar server is a framework that provides a wide range of different servers (DHCP, SMTP, FTP etc. . .). The purpose of Sambar is to allow with just one choice, to set up a complete set of services. Sambar is available for both Linux and Windows and is fully configurable through a web interface. A stripped down version of the server is provided for free and it is closed-source.

lighttpd lighttpd is small footprint web server. It is Open Source licensed under the revised BSD license. It is designed keeping in mind the memory and CPU occupation, no matter this it provides a complete set of feature that allows it to be compared with Apache [1]. lighttpd also support FastCGI [3] that is an extension to CGI designed to provide high performance without the limitations of server-dependent solutions. lighttpd is the server currently used for the development of *F*ire*F*ly.

ghttpd ghttpd is a small web server released under the GPL. It provides CGI but not FastCGI. It is available just for Linux and Unix.

*F*ire*F*ly is currently using lighttpd for various reasons: operating system independent solution, simple configuration, availability of FastCGI etc.... Being *F*ire*F*ly an AJAX application it is actually *web server independent*, because its logic is just a collection of standard HTML files and PHP sources that can be installed in whatever CGI-aware web server available.

3.3 User interface issues

The use of AJAX often raises critics about the usability level perceived by users, mainly because the UI has to be implemented using ad-hoc Javascript libraries (see [11], [9], [2]). The UIs rendered using these libraries can be non standard from a user point of view, so a particular attention must be paid in designing them. The current *F*ire*F*ly UI is implemented using YUI from Yahoo!, the idea is to simplify the UI and keep it as similar is possible to widespread operating systems: the users list and every tool in *F*ire*F*ly is rendered within a sort of windows exposing traditional controls like drag and drop on title bar and maximize/minimize icons, in the upper part there is a status bar that mimics the feature of the status bar available under most common operating systems (see Figure 2).

3.4 System life cycle

Using AJAX for implementing an highly interactive system like *F*ire*F*ly, means to carefully design the policy of distribution of the updates among the users. The typical user activity is to append sentences in a chat session; because of the architecture of the browser, when a user clicks the submission button for a new contribution, the text is immediately sent to the server that has to bounce it to every other client.

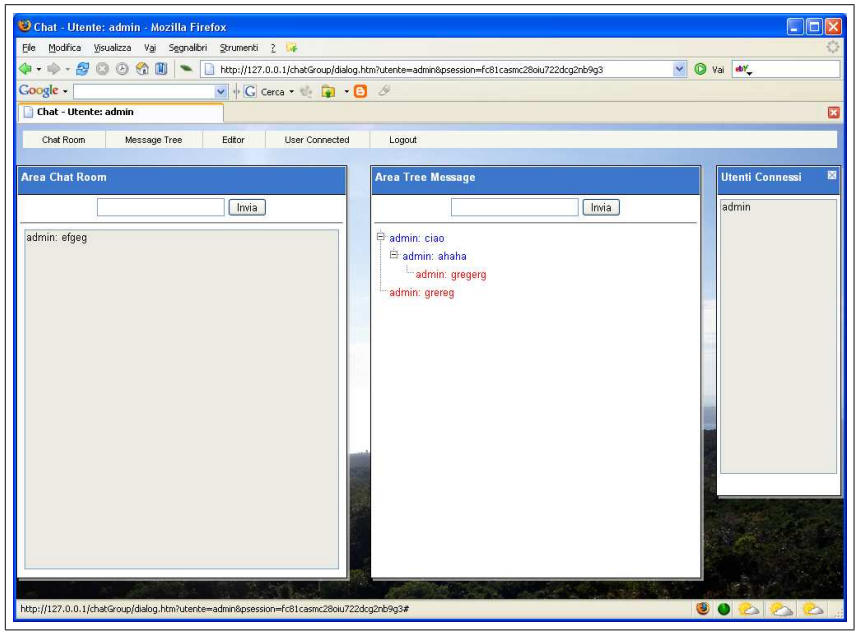

Fig. 2. A screenshot from the application.

Bouncing automatically newly available updates to the clients is not possible because the data exchange between the web server and the clients can happen just when the clients explicitly request for it. Using AJAX (and `XMLHttpRequest`) the client's browser is capable of *periodically request* updates and visualize them in the various tools. This is the key of the use of AJAX, these periodic updates cannot be avoided because of the HTTP protocol that is based on request/response mechanism [14], and the respond, that is the updates from other clients, can be sent just after a request generated by the browser. In figure 3 is shown the data exchange between the client and the server, continuous lines indicate the periodical updates, while dotted lines indicate the updates sent from client to server on every new contribution from user.

The frequency of such requests is a critical issue, too frequent requests create an heavy load on the server, while less frequent requests cause the slowdown of the interactions between users. In early testing we have used a 3 seconds interval, it is clear that this interval depends on various factors: the number of clients, the number, the size and the frequency of contributions from users. The server collects casual updates from clients

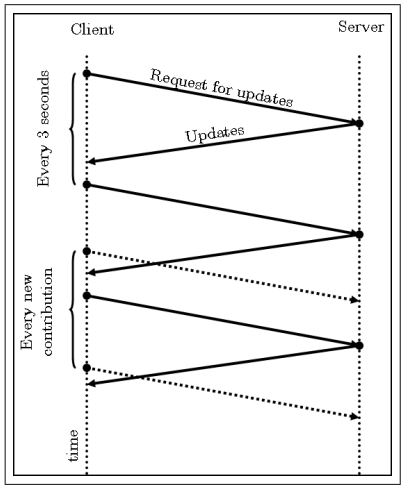

Fig. 3. Data exchange between client and server. lines between client and server indicate periodically updates, while dotted lines indicate casual updates. All the exchanged data are well-formed XML files.

and append them to the XML file that the periodical updates will request for.

4 Conclusion

The system we have designed is currently under development, our effort is in tidying up the code in order to make it modular. Our ambition is to design a modular system that would allow to developer to create tools on their own and just plug them into the system. One of the first step in this tidying up phase is to keep the XML exchange across the network the more efficient is possible, and this will request some ad-hoc measures of both generated traffic and parsing effort for both client and server.

Discussion As a summary we report here how we matched the prefixed goals reported in early sections:

- Easiness in the installation and management: *F*ire*F*ly needs just a one-step installation on the server. The installation of the web server is enough easy to be handled by common users and will not require complex settings.
- Low-end hardware: the system can be used exploiting pre-existent web server. In case no web server is available a small footprint solution can be employed.

– No Internet connection required: there is no-need for an internet connection. *F*ire*F*ly works on a LAN.
– Need for a cross platform solution: the capability of being cross-platform is achieved on both the sides, client and server, as a result of using AJAX technologies. Under various operating systems there exists plenty of web servers suitable to run *F*ire*F*ly. Whatever is the client operating system there will be a browser capable of executing the *F*ire*F*ly Javascript code.

Further undergoing developments are toward implementing new tools, like cooperative writing tool and graphical whiteboard.

References

1. Apache software fundation. http://www.apache.org/.
2. Dojo. http://dojotoolkit.org/.
3. Fastcgi. http://www.fastcgi.com/.
4. Gaztek ghttpd. http://gaztek.sourceforge.net/ghttpd/.
5. Gmail. http://www.gmail.com.
6. Html 4.01 specification: Forms. http://www.w3.org/TR/REC-html40/interact/forms.html.
7. lighttpd. http://www.lighttpd.net.
8. Sambar server. http://www.sambar.com.
9. script.aculo.us. http://script.aculo.us/.
10. Writely. http://www.writely.com.
11. Yahoo! user interface. http://developer.yahoo.com/yui/.
12. Youos. http://www.youos.com.
13. Jesse James Garrett. Ajax: A new approach to web applications. http://www.adaptivepath.com/publications/essays/archives/000385.php.
14. H. Frystyk T. Berners-Lee, R. Fielding. Hypertext transfer protocol – http/1.0. 1996. http://tools.ietf.org/html/rfc1945.