



Kaomso

www.kaomso.com

Web Applications (Thin Clients)

Modified August 17, 2004

By Scott Orsburn
© Copyright 2004, Kaomso

Introduction

This paper is intended to describe what a web-based application is. It is intended to illustrate how they are used and why they have become so popular.

Additionally, this paper will describe in detail the process taking place to run a web application without binding myself to one specific platform too much.

The two technologies referred to most will be ColdFusion from Macromedia (<http://www.macromedia.com/software/coldfusion/>), and PHP from Apache (<http://www.php.net/>). Other examples may refer to Perl or AppleScript (<http://www.apple.com/applescript/>), but conceptually, they all work the same way.

Requirements

Typically, the software we are most familiar with is simply installed on the computer, and then run. We typically refer to this type of program as desktop software or local binary. There are other terms for sure, but these pretty accurately differentiate their type from web applications. In contrast to what we conventionally think of when we refer to software, web applications require several various components to run properly. (See the section titled "Distinction" for more.) Those requirements are at least:

A web browser

More feature-rich browsers are preferable. These browsers can better leverage some of the functionality of certain web applications better than those without. Such is the case with

dynamically populated menus based on the selection made in another menu. Typically, this functionality is handled via JavaScript, therefore if your browser has a poor implementation of a JavaScript engine, then the application will probably not function correctly. It is commonplace for web application vendors to set browser requirements.

Refer to the section on JavaScript for more.

An application server

Certain server software makes a point of referring to itself with this term. ColdFusion and Apple WebObjects (<http://www.apple.com/webobjects/>) are two examples. Other examples such as PHP or Perl working in conjunction with a web server may not be as obvious an example of an application server as the earlier examples.

A network

The very concept of a web application implies the existence of a network. The more bandwidth the network can handle, then the more clients the application can handle. A faster network will also accommodate better performance of the application. Of course web app performance is dependant on the ability of the server hardware and software to handle loads too.

Distinction

Conventional software is machine executable code that resides locally on a user's computer or in a network accessible directory. When the software is executed, the machine code is loaded into the main memory of the user's computer.

When the user issues commands to the software, they are interpreted and executed locally on the user's computer.

Conversely, when a web application is executed it is done so remotely on a server computer. (At least mostly.) The physical location of the server could be most anywhere and could be invoked either on a closed network or over the Internet. The invocation of the program is done from the user's local computer and the executable code is not necessarily machine readable code, but could be interpreted code or byte code, but will be executed on the server instead of the local computer.

An exception to the remote execution of the program are technologies like JavaScript where certain code of the app is actually executed locally (on the user's machine).

The applications commands are invoked from the user's local computer and are executed on the remote server computer. Feedback from the application is sent over a network to the user's computer. For this reason, it is suggested that a faster network can lend to better application performance.

It is important to realize that input and output using a local application is very fast and never really leaves the bowels of your computer. In a thin client, all this data and communication and other goodies are being transmitted over a distance. It can take a performance hit due to all the ramifications of working over a network.

Exception

There is one notable exception to this setup. In the case of application development, the user's computer, or client computer and the server computer could be one in the same.

For all intents and purposes everything remains the same. What in effect happens is that the user is making requests to "remote" software running on itself. The server name is typically going to be the hostname localhost or the IP address 127.0.0.1.

It is important to realize that the hostname localhost and IP address 127.0.0.1 are legitimate TCP/IP designations and that network access is still being employed even in this development environment. In the networking world, it is understood that the IP address 127.0.0.1 and the domain name localhost points to the local computer.

Appeal of Web Applications

Generally speaking, there are no platform specific dependancies for the application on the client end of the application.

As noted earlier in this document, there may be some dependancies on certain software the client computer runs. But most popular platforms run a variety of software that allows most web applications to run on most any platform. As such, there is a great appeal to employ web-based software since it is not necessary for all users to run the same version of the operating system or to use similar hardware.

Kaomso holds that there are only very rare cases where platform specific or browser specific web applications are acceptable. It is our belief that by setting such requirements defeats a great advantage of using a web app in the first place. If the solution to a problem limits use of the app to a specific browser or OS another solution should be sought.

Web applications also centralize the software base. Rather than maintaining several copies of software on each user's computer, there need be only one copy of the software running on a server. This is particularly advantageous for maintenance purposes. New versions or updates can be applied once and all users immediately benefit from the change.

There may also be some savings on software licenses or software ownership grants. However, it is not wise to bank on this potential advantage. Since fewer copies of the software title are being sold, then the vendors must recoup research and development costs as well as make their profit off of fewer sales. As a result, some web applications can be quite

costly. Several million dollars is not unheard of.

The Client's Role

Conventional software has some provision for taking input from the user and for displaying output to the user. In GUI applications the application may open a window and write the output to it. A command-line interface (CLI) such as a terminal session in an operating system like Mac OS X may simply write output to the terminal.

For web applications it is convenient to leverage a web browser as the interface for a web application. By using HTML form widgets and hyperlinks, commands can be issued to the application. The application can then display output as it's fed over the network to the browser.

In fact, to the user, there is often little distinction made between a web application and a web page.

This behavior is most certainly due to the fact that most web applications use HTML form widgets, which are in turn familiar to the GUI widgets most computer users use with conventional software.

Furthermore, since the application interacts with the user a la a web browser, the application's interface looks much like a webpage.

What the browser does

The user inputs all the desired information such as choosing items from pop-up menus, or filling in text boxes. Then a command is issued to the application—usually by clicking on a button or a hyperlink.

What the browser does is generates an HTTP request to the server and waits for an answer to come back from the server. Typically, that "answer" is actually data in the form of HTML code.

The Server's Role

The server's role is typically far more

complicated than that of the client.

The term server actually refers to a combination of hardware and software. To this point the term software has referred to the web application. However, for the web application to work, the server needs to run a couple pieces of machine executable software.

It's fun to note that a proof of concept has been built to demonstrate the power of PHP. A simple web server has been built using PHP which suggests that server software does not have to be machine code.

This machine executable software is at least an HTTP server and some software to run the web application.

The HTTP server is what "listens" to the network and responds to commands being issued to the web application. The software to run the web application may be the web application itself or something like PHP or Perl.

If the web application runs itself, then it is likely the machine executable style of web application mentioned earlier in this document. These are usually developed with technologies like C++.

Usually however, most web applications are interpreted which means that additional server software will be necessary.

Several technologies have evolved to handle the demand for executing web applications. C++ and AppleScript have new libraries to extend their capabilities. And then new technologies like ASP.NET (<http://www.microsoft.com/serviceproviders/aspnethost/>), JavaScript, CFML, and PHP have made their way onto the scene.

A bit later in this section the distinction between using languages like C++ and ones like PHP should become clear.

Usually, the commands issued by the user will be requesting a file on the server. The HTTP server software is responsible for accessing that file. The HTTP server software then passes the content

of the requested file back to the client. This is the response the client computer is waiting for as noted in the “Client’s Role” section.

Where the application aspect comes in is when there is information requested by the client that is outside of the scope of what a conventional HTTP server can handle.

The requested file may have some code that needs to be run instead of just HTML code. The HTTP server will depend on the application interpreter to accommodate this aspect of the web application. This is what software like Perl and PHP does. Other common software that fills this role is technologies such as ASP and JSP.

The code that the HTTP server cannot handle is passed on to the web application server. The output it generates is passed back to the HTTP server, which passes it right on to the client.

Some web applications however, are not interpreted but rather machine executable code. The requested file by the client may call some subroutines that make up the web application. While it was pointed out earlier that these routines may be interpreted, sometimes they are not. In the case that the subroutines are not interpreted, then the web application is executed and the results are then passed back to the web server. Web applications developed with C++ for example work like this.

The effect is the same in either case, but the way the application is executed makes a difference.

JavaScript

Perhaps the strongest virtue of web applications is their platform independence, as mentioned earlier. This is because the actual execution of the application is done on a common computer instead of the user’s own computer. As a result, all clients interact with the application in a common manner.

One notable exception to this is the JavaScript technology. JavaScript was devised as a way for program or script code to be downloaded and executed in the user’s web browser.

It provides the ability to open new browser windows, write output to the window, and modifying options in form widgets, to name just some of the functionality.

Many web applications employ JavaScript as it can bring a better user experience or provide necessary functionality that may not be possible by using another technology.

The trouble with this is that JavaScript code is actually executed locally on the user’s computer. Which means that this should actually have been pointed out in the “Client’s Role” section. But, JavaScript can be used in conjunction—in a manner of speaking, with the remote portion of the application. So, it is still appropriate here too.

Here is the problem. Not all browsers have implemented a JavaScript engine the same way. As a result, some browsers have different capabilities than others.

This can have an ill affect on how well the web application works and can actually defeat some of the real advantage of employing a web application in the first place.

Fortunately, most browser vendors are trying to come to more common ground concerning the functionality of their JavaScript engines. As a result the differences between browsers and the way they run JavaScript code is largely irrelevant now.

Summary

For a web application to be used, a client passes a request to the server. This is most commonly done via a web browser.

That request is acknowledged by the server software. The web application is executed, processes the request, and may passes some data back to the user.

The client’s browser interprets that data coming back—which is usually HTML data, and shows it to the user.

The centralization of the web applications makes it appealing from an administration and deployment standpoint. Most web application technology has great provisions for accessing database managers and gathering data from databases.

