

The Wayback Machine - <https://web.archive.org/web/20060526021611/http://www.websyodian.com/websydiando...>

Web Applications in General

This section of the *Overview of Internet Technology* discusses the following subjects:

Understanding Web Applications - [Definition](#), [Advantages](#) and [Disadvantages](#)

[How Web Applications Work](#)

[Web Application Architecture](#) - [Architecture In General](#), [Interfacing](#), [Websyodian Architecture](#)

A Definition of Web Applications

Briefly stated,

A Web Application is any application that users access using a Web Browser.

However, since modern Web Browsers these days include provisions for executing Java programs, the above definition may be a bit broad: If the application merely uses the Web Browser to load a Java program that operates independently of the Web Browser, then it - in our definition - does not qualify as a "proper" Web Application. To be a "proper" Web Application in our view, an application should use the standard facilities provided by the Web Browser, and permit the application's users to take advantage of the skills they have acquired in their normal use of their Web Browser.

An important implication of this is that "proper" Web Applications use HTML to interact with the user. The dialogue with the user consists of sending him or her HTML documents to which the user can respond, e.g. by following links, filling out forms, etc. Each user action results in a new HTML document prompting the user for another action. This goes on until the user decides to stop using the application.

Some Web Applications were born as such, in other cases they come into existence by adding a Web Browser-based user interface to an existing application. This distinction, however, does not affect the application's status as a "proper" Web Application.

Web Application Advantages

Web Applications are associated with many important advantages:

Web Browsers are everywhere

By now, Web Browsers are absolutely ubiquitous: By far most corporate IT user either already has one, or can easily get their hands on one. Both major Web Browser vendors (Microsoft, Netscape) provide their browser products at no charge. Microsoft even provides its Internet Explorer pre-installed on new versions of Windows 95 and NT (although that particular practice is currently [March 1998] the subject of a legal dispute).

As a user access technology, Web Browsers are simply unsurpassed: No other technology enables users on so many different platforms to access both information and applications. - There probably isn't any widely used computing platform today for which some sort of Web Browser is not available.

Web Browsers are easy to learn and to use

As anyone who has ever tried one can testify, Web Browsers are extremely easy to use. Only very few user interface conventions need to be learned for a user to become a reasonably proficient Web Browser user:

Following a link, going back and forth in the browse history list, and perhaps a few other operations. Perhaps more importantly - this interface doesn't change from web site to web site, or from Web Application to Web Application. Once learned, the skills are applicable to any Web Application.

Web Browsers are flexible

The same Web Browser can be used to interact with many different applications. That is because Web Browsers can display almost any information. And with appropriate interpretation of the links embedded in the HTML documents, the users can perform almost any action in response to the data presented to them. When an organization introduces a new application, everybody can immediately start using it - all they have to know is the URL of an entry point to the application.

Web Browsers are powerful

At first sight it may seem that Web Browsers are limited to displaying information. That is not the case, although they do that very well indeed. Using HTML's powerful `<FORM>` feature, which allows the user to enter data into the application, Web Browsers can serve as user interface for a wide range of applications.

Web Applications are inherently Internet-enabled

Web Browsers were invented for the Internet, designed from the ground up to work well in a wide area, potentially low bandwidth environment. They allow the number of users of an application to be very large, and widely spread out geographically, accessing the application across an organizations own, internal local or wide area network, or across the publicly accessible Internet.

Web Applications are desktop platform independent

Properly designed, a Web Application can be used by users on any platform, because nothing in the design and specification of Web Technology depends on the features of any particular operating system or platform. Web Browsers are available for every widely used computing platform in the world.

Web Applications are installed on the Web Server

Or, put slightly differently, Web Applications do not have to be installed on the machines used to access them. All that must be installed is the Web Browser, and it can be used to access many different Web Applications. When a Web Application needs to be upgraded, installation occurs on the server, and - once performed - is instantly available to all users immediately thereafter.

Web Applications are excellent for casual or infrequent users

Web Applications use a "document paradigm" in its interaction with the user. This is a great advantage for applications that are used by a large number of users, especially if these users tend to use the application relatively rarely. This is often the case with Human Resource Management applications, and applications that involve users that do not work in the organization operating the application (as is the case when the users are suppliers, partners, customers, or potential customers).

The document paradigm makes it straightforward to design a "self-explanatory user interface", because the document paradigm allows the interface designers to inform the user directly of his or her options in any given context. A simple example could look something like

[Click here if you want to confirm the suggested vacation period.](#)

[Click here if you want to reject the suggested vacation period.](#)

in a Human Resource Management application with direct application access for employees.

In many application deployment scenarios, a classic graphical user interface - in the style of most Windows and Macintosh applications - is vastly inferior to the document paradigm approach of Web Applications.

Web Application Disadvantages

Unfortunately, Web Applications are also associated with a number of disadvantages:

Implementing Web Applications can be hard

When you build Web Applications, you basically create a set of documents with embedded actions (represented as HTML links and forms). Each action is supposed to produce a document presenting the user with the result of his last action. This paradigm is very powerful, but it easily leads to a very fragmented view of the application - that of the individual document.

Also, since the HTTP protocol - which underlies all Web Applications - supports no notion of *user sessions*, the load of their implementation (which many Web Applications require) is placed on the Web Application developer, increasing the amount of work he needs to do to develop a satisfactory solution.

However, using the built-in functionality of Websyidian, and AllFusion Plex's powerful inheritance mechanisms almost eliminates these difficulties. Using a model-based, declarative approach to building Web Applications off-loads considerable work on AllFusion Plex's data-model and code generation mechanism. Work that developers using other tools are left to handle manually.

There are not enough firmly established standards and inter-browser incompatibilities exist

Microsoft's and Netscape's Web Browser products do not behave identically in many circumstances, and certainly do not always present identical documents identically. This means that there is a danger that Web Application behavior may depend on the brand and version of the user's Web Browser.

Most likely, the best Web Application developers can do - given the state-of-the-art - is to work around the incompatibilities by being aware of them, and avoiding using the features that cause them. Luckily, the incompatibilities are more nuisances than show stoppers.

Web Browsers are not panacea

Not all applications lend themselves to be Web Applications. Applications that require frequent and intensive manipulation of complex data are not likely candidates for an HTML-based user interface. However, most business-specific applications do not have such requirements - they are, in fact, mostly excellent candidates for Web-based user interfaces.

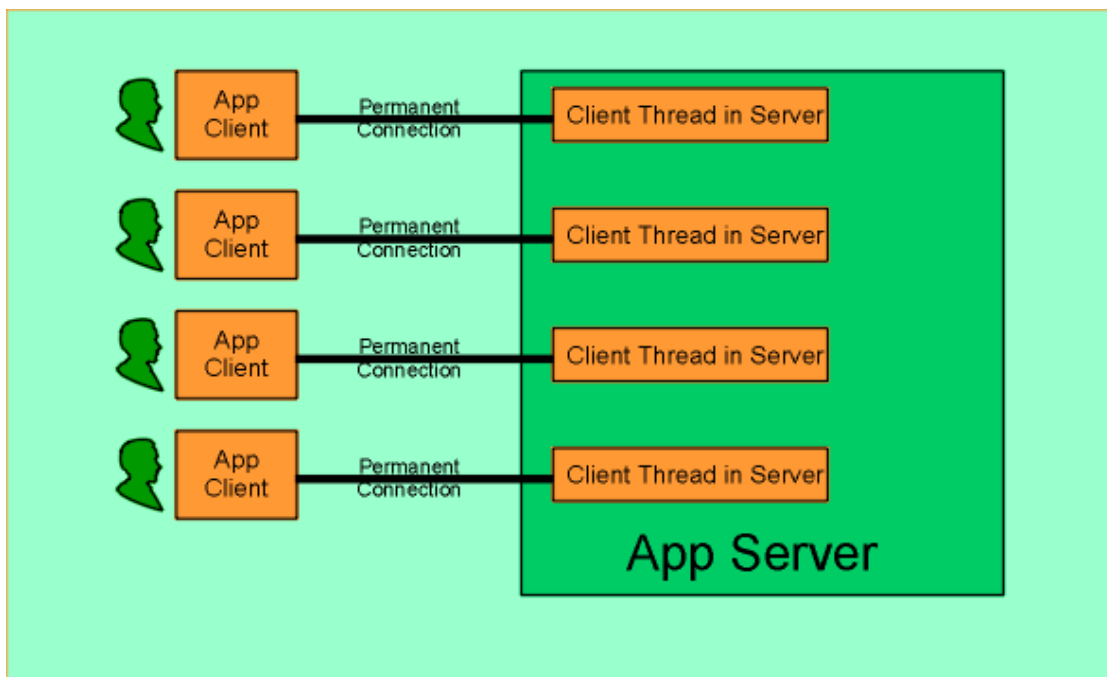
When considering the development of a Web Application, or adding a Web-based interface to an existing application, it should be carefully considered how system interaction with the user is to be accomplished using HTML.

How Web Applications Work

Web Applications are inherently client-server applications - there is always a web client and a web server operating, otherwise we would not call them Web Applications (cf. our definition [above](#)). Web applications, however, are different from traditional client-server applications in a few crucial ways.

Comparing: Classic Client-Server Architecture

To better understand the nature of Web Applications, it is useful to compare them to the "classic" client-server architecture, as illustrated by the figure below.



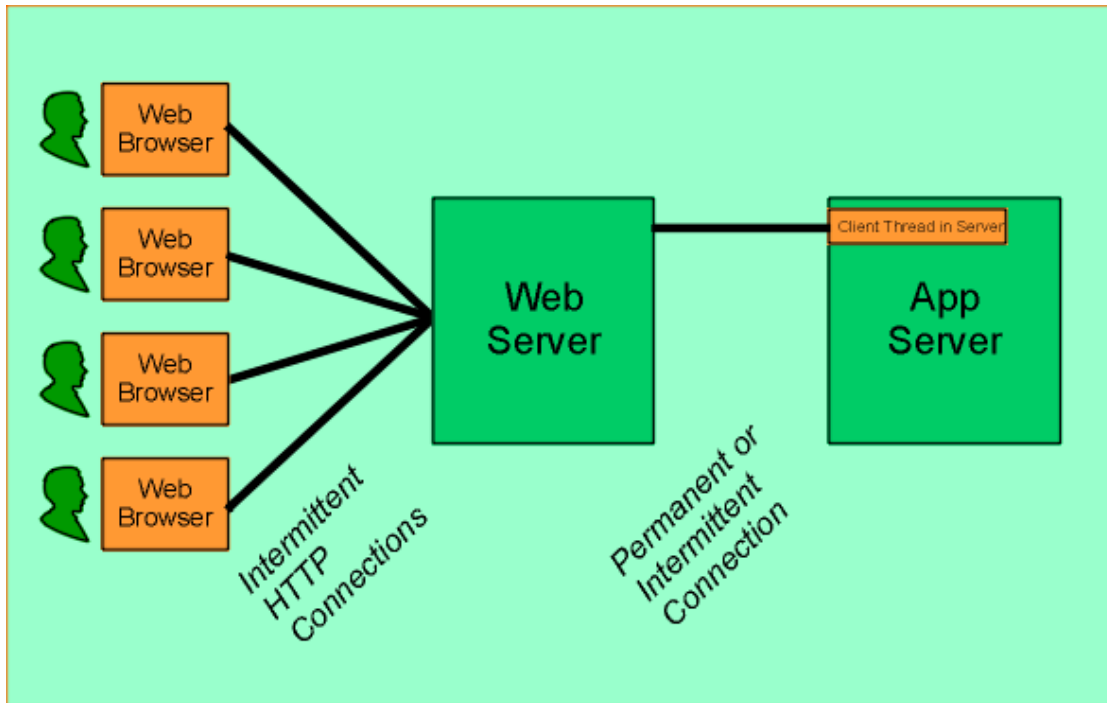
The figure above illustrates how client server applications are usually organized - each user has his own client which maintains a permanent connection to the server which maintains the state of each session by allocating a thread for each client. The thread embodies the state of the session. Applications developed using AllFusion Plex look like this - when a client function calls a server function the call itself is transferred using the permanent connection between the client and server, creating - for the duration of the server function call - a client thread in the server. You might call this approach a client-server paradigm based on *function calls*. This is how applications developed using AllFusion Plex work.

Another characteristic of classic client-server applications is that the Application Client is specific to the application, i.e. it was written to operate in that application only.

Comparing: Web Application Architecture

The architecture of Web Applications is somewhat different from that of classic client-server applications. Fundamentally, they all work by receiving requests made by users accessing the application using a Web Browser, and producing Web files (most often HTML Pages) which represent the answer to the requests received.

The basic architecture is shown below.



The figure illustrates a number of important points about Web Applications:

- A Web Browser is used as the Application Client software.

- HTML is used by the application to interact with the user.

- A Web Server (or, more precisely, a piece of Web Server Software) acts as an intermediary between the Application Client and the Application Server.

- There is a many-to-one relationship between the number of Application Clients and the Web Server software; the Web Server acts as a multiplexer/demultiplexer, routing requests from multiple Application Clients to a single Application Server and the corresponding responses back the other way, making sure that requests and responses match up correctly.

In a nutshell, Websyidian is designed to shield the software developer from the (vast majority of) differences between classic client-server applications and Web Applications. Unfortunately, it is not possible to completely shield the developer from these differences, but Websyidian attempts to do as much as possible.

Web Application Architecture

It is worthwhile to examine the architecture of Web Applications in more detail.

As mentioned in the section about [Web Technology](#), Web Server Software is free to interpret a Web Request any way it wants. Web Applications are created by developing software that makes such interpretations, and provide appropriate responses.

Web Server Software is inherently generic, so it cannot provide application specific interpretations or responses. To make Web Applications possible, Web Server Software can be configured to recognize certain Web Requests as pertaining to specific Web Applications, and route the requests to the Web Application in question.

Interfacing Web Server Software to Web Applications

The routing of Web Requests from the Web Server Software to Web Applications (and the Web Response in the opposite direction) implies the existence of an *interface* between the two. The most popular interface is called CGI - **C**ommon **G**ateway **I**nterface. It is a very simple interface, in which the Web Server Software is required to load and execute a program and encode the Web Request as string values in the execution environment of the process resulting from starting the program. The Web Response is simply written to the standard output file of the newly created process. The web server software transmits the generated output to the web browser software as HTTP packets, and the process is terminated.

There are other ways to interface to Web Applications, but CGI is the only interface supported by practically every kind of Web Server Software in existence. Most of the alternative interfaces were created to improve the performance of Web Applications, since the process creation overhead of CGI is quite substantial.

Alternative interfaces include **ISAPI** (for interfacing Microsoft's Internet Information Server to Web Applications), and **NSAPI** (for interfacing Netscape's Web Server Software offerings to Web Applications), and a number of other, less widely used interface technologies. Most of the alternatives rely on some variant of using Dynamically Loadable Libraries (DLLs) as the underlying interface technology, instead of CGI's high-overhead interface which relies on process creation.

In its initial version Websyidian only supports CGI. But, in time, Websyidian will support a wider range of Web Server Software to Web Application interfaces. However, when support for alternative interface is added, Websyidian is designed to incorporate it into existing Websyidian applications simply by adding a couple of triples, regenerating and recompiling the entire application.

Web Application Architecture in General

The architecture of various Web applications is quite diverse, so not much can be said in general about the common denominator of their architecture. The hard requirement is that a Web Application is able to retrieve a Web Request and produce a corresponding Web Response using the Web Server Software.

In the early days of the Web, most Web Applications were quite simple, consisting mostly of small programs written in scripting languages. Not much architecture there. Today, Web Application developers are much more ambitious, integrating live data from corporate databases into the Web Responses produced by their systems.

Most Web Applications developed today probably belong to the 'fat client' tradition of client-server applications, in which data sources are accessed using an SQL-based interface, which means that most of the processing required to produce a Web Response occurs on the Web Server Host.

Websyidian Application Architecture

Applications developed using Websyidian are fundamentally like other applications developed using AllFusion Plex, except that the traditional GUI-oriented client is replaced by a GUI-less client which is invoked using CGI. Instead of opening windows on a computer screen and waiting for the user to produce events, Websyidian-generated clients merely retrieve the parameters of the Web Request and call client or server functions depending on certain, Websyidian-generated 'dispatcher' values which are always included in the Web Request - Websyidian takes care of that too. The function which handles the request generate a HTML page, which is forwarded to the web user via the web server software.

The architecture of Websyidian-based applications is only bound by two things:

1. The desired functionality must be invocable using a Web Server Software to Web Application interface (limited to CGI at this point in time)
2. It must be supported by AllFusion Plex.

This leaves a virtually unlimited set of options of functionality.



Proceed with next section [Web Application Security](#).