INTRODUCTION TO A SERVER

A **server** is a system (software and suitable computer hardware) that responds to requests across a computer network to provide, or help to provide, a network service. Servers can be run on a dedicated computer, which is also often referred to as "the server", but many networked computers are capable of hosting servers. In many cases, a computer can provide several services and have several servers running.

- The term *server* is used quite broadly in information technology. Despite the many server-branded products available (such as server versions of hardware, software or operating systems), in theory any computerized process that shares a resource to one or more client processes is a server. To illustrate this, take the common example of file sharing. While the existence of files on a machine does not classify it as a server, the mechanism which shares these files to clients by the operating system is the server.

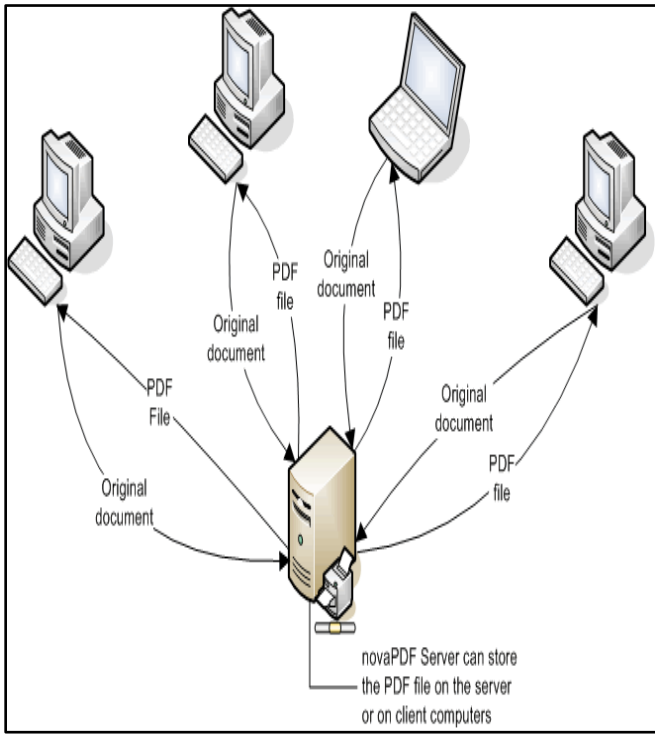
- Similarly, consider a web server application (such as the multiplatform "Apache HTTP Server"). This web server software can be *run* on any capable computer. For example, while a laptop or personal computer is not typically known as a server, they can in these situations fulfill the role of one, and hence be labeled as one. It is, in this case, the machine's role that places it in the category of server.

- In the hardware sense, the word *server* typically designates computer models intended for hosting software applications under the heavy demand of a network environment. In this server configuration one or more machines, either a computer or a computer appliance, share information with each other with one acting as a host for the other.

SERVER HRDWARE

Hardware requirement for servers vary, depending on the server application. Absolute CPU speed is not quite as critical to a server as it is to a desktop machine. Servers' duties to provide service to many users over a network lead to different requirements such as fast network connections and high I/O throughout. Since servers are usually accessed over a network, they may run in headless mode without a monitor or input device. Processes that are not needed for the server's function are not used. Many servers do not have a graphical user interface (GUI) as it is unnecessary and consumes resources that could be allocated elsewhere. Similarly, audio and USB interfaces may be omitted.

- In the client/server programming model, a server is a program that awaits and fulfills requests from client programs in the same or other computers. A given application in a computer may function as a client with requests for services from other programs and also as a *server* of requests from other programs.
- Specific to the Web, a web server is the computer program (housed in a computer) that serves requested html pages or files. A Web *client* is the requesting program associated with the user. The Web **Browser** in your computer is a client that requests HTML files from Web servers.



TYPES OF SERVER

1. Application server a server dedicated to running certain software applications.
2. Catalog server a central search point for information across a distributed network.
3. Communications server carrier-grade computing platform for communications networks.
4. Compute server, a server intended for intensive (esp. scientific) computations.
5. Database server provides database services to other computer programs or computers.
6. Fax server provides fax services for clients.
7. File server provides remote access to files.

8. Game server a server that video game clients connect to in order to play online together.
9. Home server a server for the home.
10. Mail server handles transport of and access to email.
11. Mobile Server or Server on the Go is an Intel Xeon processor based server class laptop form factor computer.
12. Name server or DNS.
13. Print server
    Uprovides printer services.
14. Proxy server acts as an intermediary for requests from clients seeking resources from other servers.
15. Sound server provides multimedia broadcasting, streaming.
16. Stand-alone server a server on a Windows network that neither belongs to nor governs a Windows domain
17. Web server a server that HTTP clients connect to in order to send commands and receive responses along with data contents

SERVER OPERATING SYSTEM

Server-oriented operating systems tend to have certain features that make them more suitable for the server environment, such as:

1. GUI not available or optional
2. Ability to reconfigure and update both hardware and software to some extent without restart
3. Advanced backup facilities to permit regular and frequent online backups of critical data
4. Transparent data transfer between different volumes or devices
5. Flexible and advanced networking capabilities
6. Automation capabilities such as daemons in UNIX and services in Windows
7. Tight system security, with advanced user, resource, data, and memory protection.

Server-oriented operating systems can, in many cases, interact with hardware sensors to detect conditions such as overheating, processor and disk failure, and consequently alert an operator or take remedial measures themselves.

DIFFERENT SIZE OF SERVER

- Rack server
- Tower server
- Miniature (home) servers
- Mini Rack server
- Blade server
- Mobile server

PHP: Server-Side Programming Languages in Web Development

Dynamic and Static Webpages:

When we want to design a website so there are two types of website pages or web pages, which is static and dynamic web pages. The static web pages take data which don't change or may change but not using any database and we can also say that static web pages don't interface with online database. But the dynamic web pages interface online database, it's all data store in database and use MySQL to retrieve all data from database on the web pages. As we can make login pages, and registration pages and many other kind of web pages.

Techniques being used in Website developing:

We use **HTML, CSS, JavaScript, AJAX, PHP, MySQL, JQuery** and many other techniques for developing web pages, to make web pages more attractive and more useful. As there are CMS (Content Management System), which are **JOOMLA, WORDPRESS, DRUPAL**. These three CMS are being commonly used in the market of website developing.

Because these three CMS made the designing of web pages more easy. And these CMS help us to design dynamic web pages.

For dynamic web pages we use CMS if we want to design web pages easily and attractive, we can also design web pages by our own thoughts using PHP and MySQL. In which age we are living it is the time to create dynamic web pages. Of course, without using HTML, CSS, JavaScript, AJAX we can't use just PHP and MySQL but the main thing is that make secure web pages which should secure data and be useful, which is done by PHP and MySQL.

Server Side Programming Language:

PHP is a language which is used as server side programming language. Programming with PHP is not a difficult job or task. Because if you learnt C/C++ programming so it will be not change for you but at some places you will feel lot of changes using of keywords as there is no any kind of data type or primitive type in PHP programming. PHP is an easy programming language for those people who learnt by heart the JavaScript client side web programming language, and fully experience on C/C++.

The first programming language which is understood by Human that is C/C++, that's why other all programming languages else they are website programming or Desktop application programming languages have same features and using methods like C/C++. We can say the PHP programming is an Object Oriented programming, because it has also inheritance, polymorphism etc. techniques.

PHP is a server scripting language, and a powerful tool for making dynamic and interactive Web pages.
PHP is a widely-used, free, and efficient alternative to competitors such as Microsoft's ASP.

Easy Learning with "Show PHP"

Our "Show PHP" tool makes it easy to learn PHP, it shows both the PHP source code and the HTML output of the code.

Example

```
<!DOCTYPE html>
<html>
<body>

<?php
echo "My first PHP script!";
?>

</body>
</html>
```

What is PHP?

- PHP is an acronym for "PHP: Hypertext Preprocessor"
- PHP is a widely-used, open source scripting language
- PHP scripts are executed on the server
- PHP is free to download and use

PHP is an amazing and popular language!

It is powerful enough to be at the core of the biggest blogging system on the web (WordPress)!
It is deep enough to run the largest social network (Facebook)!
It is also easy enough to be a beginner's first server side language!

What is a PHP File?

- PHP files can contain text, HTML, CSS, JavaScript, and PHP code
- PHP code are executed on the server, and the result is returned to the browser as plain HTML
- PHP files have extension ". php"

What Can PHP Do?

- PHP can generate dynamic page content
- PHP can create, open, read, write, delete, and close files on the server
- PHP can collect form data
- PHP can send and receive cookies
- PHP can add, delete, modify data in your database
- PHP can be used to control user-access
- PHP can encrypt data

With PHP you are not limited to output HTML. You can output images, PDF files, and even Flash movies. You can also output any text, such as XHTML and XML.

Why PHP?

- PHP runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.)
- PHP is compatible with almost all servers used today (Apache, IIS, etc.)
- PHP supports a wide range of databases
- PHP is free. Download it from the official PHP resource: www.php.net
- PHP is easy to learn and runs efficiently on the server side

What Do I Need?

To start using PHP, you can:

- Find a web host with PHP and MySQL support
- Install a web server on your own PC, and then install PHP and MySQL

Use a Web Host With PHP Support

If your server has activated support for PHP you do not need to do anything. Just create some. php files, place them in your web directory, and the server will automatically parse them for you.
You do not need to compile anything or install any extra tools.
Because PHP is free, most web hosts offer PHP support.

Set Up PHP on Your Own PC

However, if your server does not support PHP, you must:

- install a web server
- install PHP
- install a database, such as MySQL

The official PHP website (PHP.net) has installation instructions for PHP: http://php.net/manual/en/install.php

Basic PHP Syntax

A PHP script can be placed anywhere in the document.
A PHP script starts with **<?php** and ends with **?>**:

```
<?php
// PHP code goes here
?>
```

The default file extension for PHP files is ".php".
A PHP file normally contains HTML tags, and some PHP scripting code.
Below, we have an example of a simple PHP file, with a PHP script that uses a built-in PHP function "echo" to output the text "Hello World!" on a web page:

Example

```
<!DOCTYPE html>
<html>
<body>
<h1>My first PHP page</h1>
<?php
echo "Hello World!";
?>
</body>
</html>
```

Output; hello word

Creating (Declaring) PHP Variables

In PHP, a variable starts with the $ sign, followed by the name of the variable:

Example

```
<?php
$txt = "Hello world!";
$x = 5;
$y = 10.5;
?>
```

After the execution of the statements above, the variable **$txt** will hold the value **Hello world!**, the variable **$x** will hold the value **5**, and the variable **$y** will hold the value **10.5**.

Note: When you assign a text value to a variable, put quotes around the value.

Note: Unlike other programming languages, PHP has no command for declaring a variable. It is created the moment you first assign a value to it.

Application programming interface

An **application programming interface** (**API**) is a specification intended to be used as an interface by software components to communicate with each other. An API may include specifications for routines, data structures, object classes, and variables. An API specification can take many forms, including an International Standard such as POSIX or vendor documentation such as the Microsoft Windows API, or the libraries of a programming language, e.g. Standard Template Library in C++ or Java API.

An API differs from an application binary interface (ABI) in that the former is source code based while the latter is a binary interface. For instance POSIX is an API, while the Linux Standard Base is an ABI.[1]

Language used

An **API** can be:

- language-dependent, meaning it is only available by using the syntax and elements of a particular language, which makes the API more convenient to use.
- language-independent, written so that it can be called from several programming languages. This is a desirable feature for a service-oriented API that is not bound to a specific process or system and may be provided as remote procedure calls or web services. For example, a website that allows users to review local restaurants is able to layer their reviews over maps taken from Google Maps, because Google Maps has an API that facilitates this functionality. Google Maps' API controls what information a third-party site can use and how they can use it.

The term API may be used to refer to a complete interface, a single function, or even a set of APIs provided by an organization. Thus, the scope of meaning is usually determined by the context of usage.

Detailed explanation

An API may describe the ways in which a particular task is performed. In procedural languages like C language the *action* is usually *mediated* by a function call. Hence the API usually includes a description of all the functions/routines it provides. For instance: the `math.h` include file for the C language contains the definition of the function prototypes of the mathematical functions available in the C language library for mathematical processing (usually called `libm`). This file describes how to *use* the functions included in the given library: the function prototype is a signature that describes the number and types of the parameters to be passed to the functions and the type of the return value. The *behavior* of the functions is usually described in more details in a human readable format in printed books or in electronic formats like the man pages: e.g. on Unix systems the command `man 3 sqrt` will present the signature of the function `sqrt` in the form:

```
SYNOPSIS
            #include <math.h>
            double sqrt(double X);
            float  sqrtf(float X);
DESCRIPTION
       DESCRIPTION
       sqrt computes the positive square root of the argument. ...
RETURNS
       On success, the square root is returned. If X is real and
positive...
```

That means that the function returns the square root of a positive floating point number (`single` or `double` precision) as another floating point number. Hence the API in this case can be interpreted as the collection of the

include files used by the C language and its human readable description provided by the man pages.

Documentation

Many program development environments provide the documentation associated with an API in some digital format, e.g. perl comes with the tool perldoc:

```
$ perldoc -f sqrt
       sqrt EXPR
       sqrt    #Return the square root of EXPR.  If EXPR is omitted,
returns
               #square root of $_.  Only works on non-negative
operands, unless
               #you've loaded the standard Math::Complex module.
```

python comes with the tool pydoc:

```
$ pydoc math.sqrt
Help on built-in function sqrt in math:
math.sqrt = sqrt(...)
    sqrt(x)
    Return the square root of x.
```

ruby comes with the tool `ri`:

```
$ ri Math::sqrt
------------------------------------------------------------ Math::sqrt
     Math.sqrt(numeric)    => float
------------------------------------------------------------------------
     Returns the non-negative square root of _numeric_.
```

Java comes with the documentation organized in HTML pages (JavaDoc format), while Microsoft distributes the API documentation for its languages (Visual C++, C#, Visual Basic, F#, etc...) embedded in Visual Studio's help system.

API in object-oriented languages

In object-oriented languages, an API usually includes a description of a set of class definitions, with a set of behaviors associated with those classes. This abstract concept is associated with the real functionality exposed, or made available, by the classes that are implemented in terms of class methods (or more generally by all its public components hence all public methods, but also possibly including any internal entity made public, like fields, constants, nested objects, enums...).

The API in this case can be conceived as the totality of all the methods publicly exposed by the classes (usually called the class *interface*). This means that the API prescribes the methods by which one interacts with/handles the objects derived from the class definitions.

More generally, one can see the API as the collection of all the *kinds* of objects one can derive from the class definitions, and their associated possible behaviors. Again: the use is mediated by the public methods, but in this interpretation, the methods are seen as a *technical detail* of how the behavior is implemented.

For instance: a class representing a `Stack` can simply expose publicly two methods `push()` (to add a new item to the stack), and `pop()` (to extract the last item, ideally placed on top of the stack).

In this case the API can be interpreted as the two methods `pop()` and `push()`, or, more generally, as the *idea* that one can use an item of type `Stack` that implements the behavior of a stack: a pile *exposing* its top to add/remove elements. The second interpretation appears more appropriate in the spirit of object orientation.

This concept can be carried to the point where a class interface in an API has no methods at all, but only behaviors associated with it. For instance, the Java language and Lisp (programming language) API include the interface `Serializable`, which is a marker interface that requires that each class that implements it should behave in a serialized fashion. This does not require to have any public method, but rather requires that any class that implements it to have a representation that can be *saved* (serialized) at any time (this is typically true for any class containing simple data and no link to external resources, like an open connection to a file, a remote system, or an external device).

Similarly the behavior of an object in a concurrent (multi-threaded) environment is not necessarily determined by specific methods, belonging to the interface implemented, but still belongs to the API for that Class of objects, and should be described in the documentation.[2]

In this sense, in object-oriented languages, the API defines a set of object behaviors, possibly mediated by a set of class methods.

In such languages, the API is still distributed as a library. For example, the Java language libraries include a set of APIs that are provided in the form of the JDK used by the developers to build new Java programs. The JDK includes the documentation of the API in JavaDoc notation.

The quality of the documentation associated with an API is often a factor determining its success in terms of ease of use.

API libraries and frameworks

An API is usually related to a software library: the API describes and prescribes the *expected behavior* while the library is an *actual implementation* of this set of rules. A single API can have multiple implementations (or none, being abstract) in the form of different libraries that share the same programming interface.

An API can also be related to a software framework: a framework can be based on several libraries implementing several APIs, but unlike the normal use of an API, the *access* to the behavior *built into the framework* is mediated by extending its content with new classes plugged into the framework itself. Moreover the overall program flow of control can be out of the control of the caller, and in the hands of the framework via inversion of control or a similar mechanisms.[3]

API and protocols

An API can also be an implementation of a protocol.

In general the difference between an API and a protocol is that the protocol defines a standard way to exchange requests and responses based on a common transport and agreeing on a data/message exchange format, while an API (not implementing a protocol) is usually implemented as a library to be used directly: hence there can be no *transport* involved (no information physically transferred from/to some remote machine), but rather only simple information exchange via *function calls* (local to the machine where the elaboration takes place) and data is exchanged in formats expressed in a specific language.[4]

When an API implements a protocol it can be based on proxy methods for remote invocations that underneath rely on the communication protocol. The role of the API can be exactly to hide the detail of the transport protocol. E.g.: RMI is an API that implements the JRMP protocol or the IIOP as RMI-IIOP.

Protocols are usually shared between different technologies (system based on given computer programming languages in a given operating system) and usually allow the different technologies to exchange information, acting as an abstraction/mediation level between the two *worlds*. While APIs can be specific to a given technology: hence

the APIs of a given language cannot be used in other languages, unless the function calls are wrapped with specific adaptation libraries.

Object API and protocols

An object API can prescribe a specific object exchange format, an object exchange protocol can define a way to transfer the same kind of information in a message sent to a remote system.

When a message is exchanged via a protocol between two different platforms using objects on both sides, the object in a programming language can be transformed (marshalled and unmarshalled) in an object in a remote and different language: so, e.g., a program written in Java invokes a service via SOAP or IIOP written in C# both programs use APIs for remote invocation (each locally to the machine where they are working) to (remotely) exchange information that they both convert from/to an object in local memory.

Instead when a similar object is exchanged via an API local to a single machine the object is effectively exchanged (or a reference to it) in memory: e.g. via the memory allocated by a single process, or among multiple processes using shared memory or other sharing techologies like tuple spaces.

API sharing and reuse via virtual machine

Some languages like those running in a virtual machine (e.g. .NET CLI compliant languages in the Common Language Runtime and JVM compliant languages in the Java Virtual Machine) can share APIs.

In this case the virtual machine enables the language interoperation thanks to the common denominator of the virtual machine that abstracts from the specific language using an intermediate bytecode and its language binding.

Hence this approach maximizes the code reuse potential for all the existing libraries and related APIs.

Web APIs

When used in the context of web development, an API is typically defined as a set of Hypertext Transfer Protocol (HTTP) request messages, along with a definition of the structure of response messages, which is usually in an Extensible Markup Language (XML) or JavaScript Object Notation (JSON) format. While "Web API" is virtually a synonym for web service, the recent trend (so-called Web 2.0) has been moving away from Simple Object Access Protocol (SOAP) based services towards more direct Representational State Transfer (REST) style communications.[5] Web APIs allow the combination of multiple services into new applications known as mashups.[6]

Web use to share content

The practice of publishing APIs has allowed web communities to create an open architecture for sharing content and data between communities and applications. In this way, content that is created in one place can be dynamically posted and updated in multiple locations on the web.

1. Photos can be shared from sites like Flickr and Photobucket to social network sites like Facebook and MySpace.
2. Content can be embedded, e.g. embedding a presentation from SlideShare on a LinkedIn profile.
3. Content can be dynamically posted. Sharing live comments made on Twitter with a Facebook account, for example, is enabled by their APIs.
4. Video content can be embedded on sites which are served by another host.
5. User information can be shared from web communities to outside applications, delivering new functionality to the web community that shares its user data via an open API. One of the best examples of this is the Facebook Application platform. Another is the Open Social platform.[7]

Implementations

The POSIX standard defines an API that allows a wide range of common computing functions to be written in a way such that they may operate on many different systems (Mac OS X, and various Berkeley Software Distributions (BSDs) implement this interface); however, making use of this requires re-compiling for each platform. A compatible API, on the other hand, allows compiled object code to function without any changes to the system implementing that API. This is beneficial to both software providers (where they may distribute existing software on new systems without producing and distributing upgrades) and users (where they may install older software on their new systems without purchasing upgrades), although this generally requires that various software libraries implement the necessary APIs as well.

Microsoft has shown a strong commitment to a backward compatible API, particularly within their Windows API (Win32) library, such that older applications may run on newer versions of Windows using an executable-specific setting called "Compatibility Mode".[8]

Apple Inc. has shown less concern, breaking compatibility or implementing an API in a slower "emulation mode"; this allows greater freedom in development, at the cost of making older software obsolete.

Among Unix-like operating systems, there are many related but incompatible operating systems running on a common hardware platform (particularly Intel 80386-compatible systems). There have been several attempts to standardize the API such that software vendors may distribute one binary application for all these systems; however, to date, none of these have met with much success. The Linux Standard Base is attempting to do this for the Linux platform, while many of the BSD Unixes, such as FreeBSD, NetBSD, and OpenBSD, implement various levels of API compatibility for both backward compatibility (allowing programs written for older versions to run on newer distributions of the system) and cross-platform compatibility (allowing execution of foreign code without recompiling).

Release policies

The two options for releasing API are:

1. Protecting information on APIs from the general public. For example, Sony used to make its official PlayStation 2 API available only to licensed PlayStation developers. This enabled Sony to control who wrote PlayStation 2 games. This gives companies quality control privileges and can provide them with potential licensing revenue streams.
2. Making APIs freely available. For example, Microsoft makes the Microsoft Windows API public, and Apple releases its APIs Carbon and Cocoa, so that software can be written for their platforms.

A mix of the two behaviors can be used as well.

APIs and Copyrights

In 2010 Oracle sued Google for having distributed a new implementation of Java embedded in the Android operating system.[9] The litigation concerns also the fact that Google did not acquire any permission to reproduce the Java API. Instead a similar permission was given to the OpenJDK project. However, Hon. William Alsup ruled on May 31, 2012, that APIs cannot be copyrighted.

API examples

- ASPI for SCSI device interfacing
- Carbon and Cocoa for the Macintosh
- DirectX for Microsoft Windows
- EHLLAPI

- Java APIs
- ODBC for Microsoft Windows
- OpenAL cross-platform sound API
- OpenCL cross-platform API for general-purpose computing for CPUs & GPUs
- OpenGL cross-platform graphics API
- OpenMP API that supports multi-platform shared memory multiprocessing programming in C, C++ and Fortran on many architectures, including Unix and Microsoft Windows platforms.
- Simple DirectMedia Layer (SDL)
- Talend integrates its data management with BPM from Bonita Open Solution

Language bindings and interface generators

APIs that are intended to be used by more than one high-level programming language often provide, or are augmented with, facilities to automatically map the API to features (syntactic or semantic) that are more natural in those languages. This is known as language binding, and is itself an API. The aim is to encapsulate most of the required functionality of the API, leaving a "thin" layer appropriate to each language.

Below are listed some interface generator tools which bind languages to APIs at compile time.

- SWIG open-source interfaces bindings generator from many languages to many languages (Typically Compiled->Scripted)
- F2PY:[10] Fortran to Python interface generator.

References

[1] Stoughton, Nick (April 2005). "Update on Standards" (https://db.usenix.org/publications/login/2005-04/openpdfs/standards2004.pdf) (PDF). USENIX. . Retrieved 2009-06-04.

[2] Bloch, Joshua (2008). "Effective Java (2nd edition)" (http://java.sun.com/docs/books/effective/). Addison-Wesley. pp. 259–312. ISBN 978-0-321-35668-0. .

[3] Fowler, Martin. "Inversion Of Control" (http://martinfowler.com/bliki/InversionOfControl.html). .

[4] "API vs Protocol" (http://c2.com/cgi/wiki?ApiVsProtocol). .

[5] Benslimane, Djamal; Schahram Dustdar, and Amit Sheth (2008). "Services Mashups: The New Generation of Web Applications" (http://dsonline.computer.org/portal/site/dsonline/menuitem.9ed3d9924aeb0dcd82ccc6716bbe36ec/index.jsp?&pName=dso_level1&path=dsonline/2008/09&file=w5gei.xml&xsl=article.xsl). *IEEE Internet Computing, vol. 12, no. 5*. Institute of Electrical and Electronics Engineers. pp. 13–15. .

[6] Niccolai, James (2008-04-23), "So What Is an Enterprise Mashup, Anyway?" (http://www.pcworld.com/businesscenter/article/145039/so_what_is_an_enterprise_mashup_anyway.html), *PC World*,

[7] "Dynamic Community content via APIs". October 26, 2009.

[8] Microsoft (October 2001). "Run Older Programs On Windows XP" (http://www.microsoft.com/windowsxp/using/helpandsupport/learnmore/appcompat.mspx). Microsoft. p. 4. .

[9] "Oracle and the End of Programming As We Know It" (http://www.drdobbs.com/jvm/232901227). DrDobbs. 2012-05-01. . Retrieved 2012-05-09.

[10] "F2PY.org" (http://www.f2py.org/). F2PY.org. . Retrieved 2011-12-18.

External links

- What is an API? Your Guide to the Internet (R)evolution (http://www.3scale.net/wp-content/uploads/2012/06/What-is-an-API-1.0.pdf)
- How to design a good API and why it matters (http://lcsd05.cs.tamu.edu/slides/keynote.pdf)
- How to Write an API (http://www.lior.ca/publications/api_design.pdf)
- LMAX Application Programming Interface (API) technology (http://www.lmax.com/trading-tech/api-trading)

Article Sources and Contributors

Application programming interface *Source*: http://en.wikipedia.org/w/index.php?oldid=498937988 *Contributors*: 213.121.101.xxx, 24.108.233.xxx, 24.93.53.xxx, 4483APK, 64.105.112.xxx, 90, Aa1bb2cc3dd4ee5, AaronL, Aarsalankhalid, AbdulKhaaliq2, Adah, Addshore, Ae-a, Aeons, Aeternus, Ahunt, Ahzahraee, Airplaneman, Alan d, Alex43223, Altaf.attari86, Altenmann, Amanuse, Ancheta Wis, Andre Engels, Andremi, Andres, Andrey86, Andy16666, Anteru, Apoltix, Arindra r, Arjayay, Arthur Davies Sikopo, Aruton, Ashamerie, Asydwaters, Atheken, Atreys, Aude, Auntof6, Avk15gt, Awg1010, Bamyers99, Bdesham, Beano, Bearcat, Betterusername, Bevo, Bhat sudha, Bikingviking, Blackcats, Bobo192, Boing! said Zebedee, Bookiewookie, Borgx, Boyprose, Brianski, Bryan Derksen, BryanG, Bryanmonroe, CYD, Calton, CanisRufus, Capricorn42, Chadsmith729, Chameleon, Chealer, Chicago god, Chiefcoolbreeze, Chituokol1, ClaudiaHetman, CloudNine, Colonoh, Consult.kirthi, Conversion script, Coolbloke94, Courcelles, Cybercobra, Cynical, DShantz, Damian Yerrick, Daniduc, Danja, Darklight, Davejohnsan, Davemck, David Gerard, Davron, Dawidl, DeeKay64, Deepugn, Dennislees, Derek farn, Detnos, Diego Moya, Diomidis Spinellis, Dipskinny, Discospinster, Dmarquard, Download, Dqpeck, Dr Marcus Hill, Dreadstar, Drewmeyers, Dschach, Dsnelling, Dylan620, EVula, Ebessman, Ed Poor, Edcolins, Edwardkerlin, Efa, Egmontaz, Ehn, Elf, Ellmist, Eloquence, Enochlau, Enric Naval, Epbr123, Eric Agbozo, Espoo, Excirial, Farrwill, Fieldday-sunday, Fitch, Frap, Freakimus, Frecklefoot, FrummerThanThou, Funvill, Fæ, GRAHAMUK, Gak, GeoffPurchase, Giftlite, Graham87, Greensburger, Griznant, HUB, Hadal, Harryboyles, Hashar, HenkeB, Heraclius, Hfastedge, Hmains, Humu, Husond, InShaneee, Infinitycomeo, Itai, Ivan007, Ixfd64, Izno, JHunterJ, JLaTondre, JWSchmidt, Jakuzem, JamesMLane, Jbolden1517, Jengod, Jerryobject, Jesse V., Jidanni, Jitse Niesen, Jleedev, Jmclaury, JoeB34, John of Reading, John.n-irl, JohnBlackburne, JulesH, Julien, Julienj, K12u, Kbolino, Kernel Saunters, Kichik, Kickin' Da Speaker, Kim Bruning, King of Hearts (old account 2), KnowledgeOfSelf, Kocio, Kurdo777, Landon1980, Lavers, Lee Daniel Crocker, Lekshmann, Leoholbel, Liempt, Limbo socrates, Liorma, LizardJr8, Lockeownzj00, Looc4s, Lord Chamberlain, the Renowned, Loren.wilton, Loshsu, Lotje, Lupin, M5, Mac, Mange01, Manop, Martyn Lovell, Marudubshinki, Materialscientist, Mattknox, Mav, Mbeychok, Meldraft, Mflore4d, Michael Hardy, Michael Hodgson, Michaelas10, Miguel, Mihai cartoaje, Miketwardos, Minghong, Minirogue, Miohtama, MoreThanMike, Morg, Mountain, Mpbaumg, MrOllie, Mrh30, Mshivaram.ie22, Mwarf, Myc2001, Nanshu, Neelix, NewEnglandYankee, Nopetro, Notinasnaid, Nsda, Obradovic Goran, Ocean Shores, Ohspite, Old Guard, OlegMarchuk, Oulrij, Oxymoron83, Pascal.Tesson, PaymentVision, Peak, Pearll's sun, Pengo, Pepe.agell, Percede, Pgimeno, Philip Trueman, Phuzion, Piano non troppo, Plamka, Poweroid, Prari, Pwforaker, Queenmomcat, Quinet, Quux, Qwertyus, Qx2020, Qxz, R'n'B, RJHall, Rackspacecloud, Raise exception, Randomalious, RedWolf, Redlentil, Reemrevnivek, Renatko, Rich Farmbrough, RichMorin, Riluve, Rktur, Robert K S, Robneild, Rocastelo, Ronocdh, RoyBoy, Rs rams, Rudyray, Rwwww, SERIEZ, SQGibbon, ST47, Sakhal, Salvatore Ingala, Sam Korn, Scohil, Scott Ritchie, Seth Nimbosa, Sfmontyo, Shlomif, SimonTrew, SirSandGoblin, Sj, Skeejay, Skysmith, Slady, Slurrymaster, SocialRadiusOly, Sodium, Soumyasch, Spencer, Spikey, SqueakBox, Stephenchou0722, SteveBaker, Steven J. Anderson, Suruena, Syvanen, Szajd, T-borg, Ta bu shi da yu, Tantrumizer, TastyPoutine, Tedickey, Teryx, Teutonic Tamer, The Anome, The Thing That Should Not Be, TheSoundAndTheFury, TheresaWilson, Thiotimoline, Thisisborin9, Tide rolls, Tijuana Brass, Tony1, Torchiest, Transpar3nt, TrentonLipscomb, Troymccluresf, Tseay11, Uriyan, Utype, Uzume, Varworld, Vikramtheone, Visvadinu, Vkorpor, Voidxor, WTRiker, WaltBusterkeys, Wapcaplet, Wavelength, Whatsnxt, Whitehorse212, WikHead, Willy-os, Wjejskenewr, Wysprgr2005, Xqt, Yaris678, YordanGeorgiev, Yurik, Zachlipton, Zeno Gantner, Zhinz, ZimZalaBim, Zodon, 계정명뭘로하지, 771 anonymous edits

License