Lecture 2: The Client-Server Model, Interfaces, and CGI

(01-29-2020)

(Reading: Lecture Notes)

Lecture Outline

1. Main components of a client-server application
   (1) The interface
   (2) The client and server
2. Three views of client-server model
   (1) Viewing a client-server application from different perspectives
   (2) Application's point of view
   (3) Programmer's point of view: clients
   (4) Programmer's point of view: servers
   (5) System's point of view
3. Format of client requests, their processing, CGI and CGI programming
   (1) Format of client requests
   (2) CGI and CGI programming
4. DBMS SPIs
   (1) Database languages
   (2) Database programming languages
   (3) Host programming languages with embedded database APIs
   (4) Oracle PL/SQL
   (5) Oracle Pro* C/C++ precompiler
   (6) SQLJ
   (7) JDBC
   (8) ODBC
   (9) OCI (Oracle call interface)
   (10) PHP
   (11) XML
5. Perl vs PHP
   (1) Commonality
   (2) Uniqueness

1. Main components of the client-server model

(1) The interface. It contains conventions (protocols) that regulates how clients and servers interact each other. If needed, the interface also contains common type definitions, constants, and other objects.

(2) The client and server. They differ from each other significantly due to their differences in functionalities.

a. Example 1: web-based applications

(a) Interface. The interface in all web-based applications are standard HTTP protocols. This protocol completely specifies the methods used between clients and servers.

(b) The clients can be any web browsers such as Netscape Navigator, Microsoft Internet Explorer, or even terminal based Lynx.

(c) Servers in such applications can be any web application servers.

Note: this is a special type of client-server applications. The client development part is minimal and major efforts are on server developments. This class will emphasize this class of client-server applications and focus on its server developments.

b. Example 2: SUN RPC and DCE RPC.

(a) Interface. Interfaces are written in SUN XDR (external data representation) language and DCE RPC IDL (interface definition language). Interface files are compiled with the supported interface compilers (*rpcgen* for SUN RPC and *idl* for DCE RPC).

(b) Clients and server can be developed separately once the interface is completely specified with the SUN RPC or DCE RPC library.

2. Three views of the client-server model

(1) A client-server model based application can be perceived differently from different perspectives.

(2) Application's (user's) point of view (Fig.9)

a. Characteristics: users have a much simplified view of the model. All the complicated details of networking communications and client/server implementations are transparent to them.

b. Users are presented with a user interface, which may be command-line based, normal GUI based, or web based. This interface is the window through which the user will interact with the server.

c. Note: the client-server interface as discussed previously is transparent to users. That interface is mainly meaningful to programmers/developers.

d. As the entire service is perceived by users through the user interfaces, the quality of the interfaces is very important to users. Users have to be familiar with the syntax and usage requirements of the interfaces in order to utilize the services effectively. Designing high quality user interface involves, among others, human factor knowledge.

e. However, the quality of the entire services depends heavily on the quality of the client and the server implementations, and on the quality of the networking software and hardware support.

f. Users normally are aware of the existence of the computer networks and some information about the server (IP name or address of the server host). For instance they may have to know (or may not have to know, depending upon specific servers) the port number of the server.

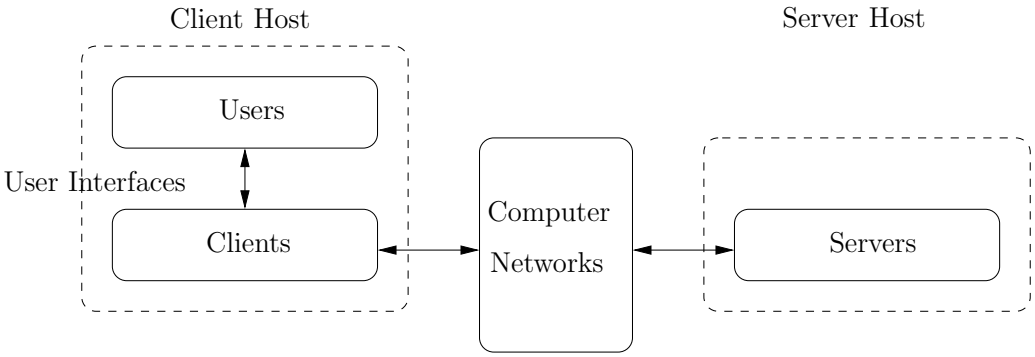


Figure 9: Client-server model from user's point of view

(3) Programmer's point of view (Fig.10): clients

a. A client can be perceived by a programmer as consisting of three portions: the client itself, the middleware, and the OS kernel.

b. Properties of clients.

(a) A client is mainly characterized by its client-server interface (eg. an *ftp* client, an *http* client, and a *telnet* client).

(b) Clients are also characterized by user interfaces they provide. There are at least three kind of popular user interfaces:

i. Traditional text-based user interfaces

ii. Traditional GUI-based user interfaces

iii. Web-based user interfaces

c. The focus of application programmers on the client side are the *client programming interfaces* (CPI). There are several different kinds of CPIs.

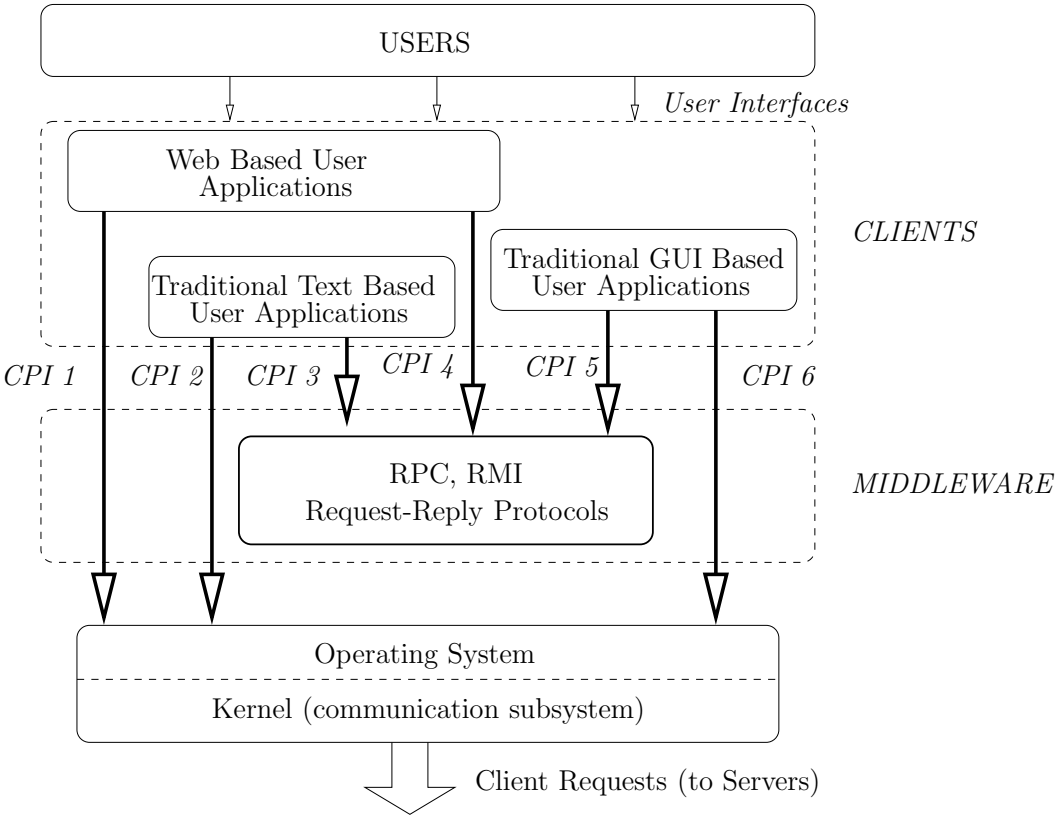


Figure 10: From programmer's viewpoint: clients in client-server model

d. CPIs.

   (a) CPIs are APIs (application programming interfaces).

   (b) CPIs are facilities (functions and system calls) that are provided by operating systems (through various programming languages) to utilize various system resources.

   (c) The nature of an CPI depends on: i. The programming language used; ii. The involvement of the *middleware* layer.

e. The *middleware* layer.

   (a) This is an layer that is intended to provide higher-level APIs to the clients and servers. Middleware layer is present in both client and server sides.

   (b) Typical middleware layer services are provided through: i. service processes (RPC and RMI services are typical examples); ii. run-time library support (RPC and JDBC drivers).

(c) As can be seen, the middleware is not essential. However, availability of a quality middleware layer relieves programmers from lower-level design tasks and hence facilitate client-server application developments.

f. A service request from a client will eventually transmitted by the client's kernel as client request to the servers.

(a) The format of these client requests is specified in the client-server interface.

(b) The server must first interpret the service requests.

(4) Programmer's point of view (Fig.11): servers

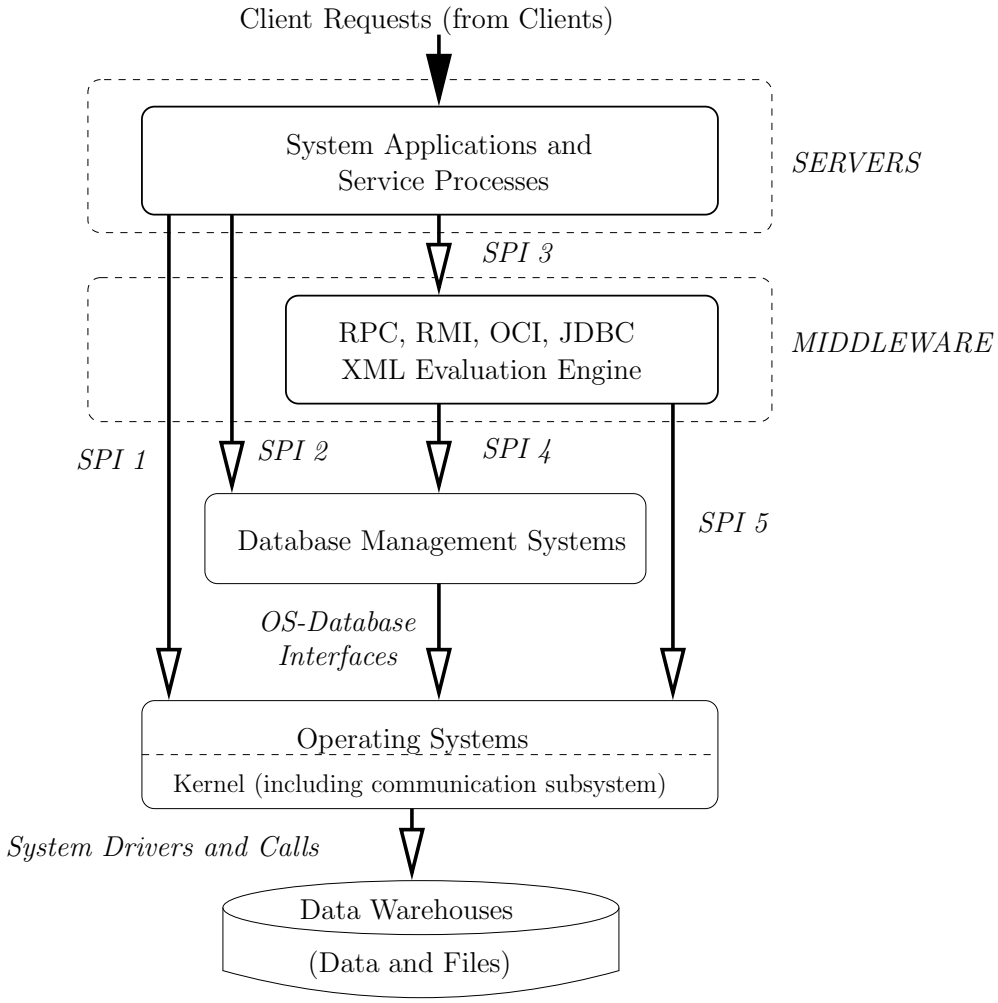


Figure 11: From programmer's viewpoint: servers in client-server model

a. A server can be perceived by a programmer as consisting of four portions: the server itself, the middleware, a DBMS, and the OS kernel.

b. Properties of servers

(a) Like clients, a server is also mainly characterized by its client-server interface (such as an *ftp* server).

(b) Another important property is whether the server is multi-threaded or single-threaded (also called *concurrent* vs *iterative*).

c. An DBMS in many modern Internet servers is essential. DBMSs are unique in servers. Clients normally do not interact with DBMSs directly.

d. SPIs

(a) Like CPIs, SPIs are APIs.

(b) SPIs are the main tools that can be used by programmers to implement servers.

(c) The presence of an DBMS provides another class of SPIs, which allow a server to interact with database systems.

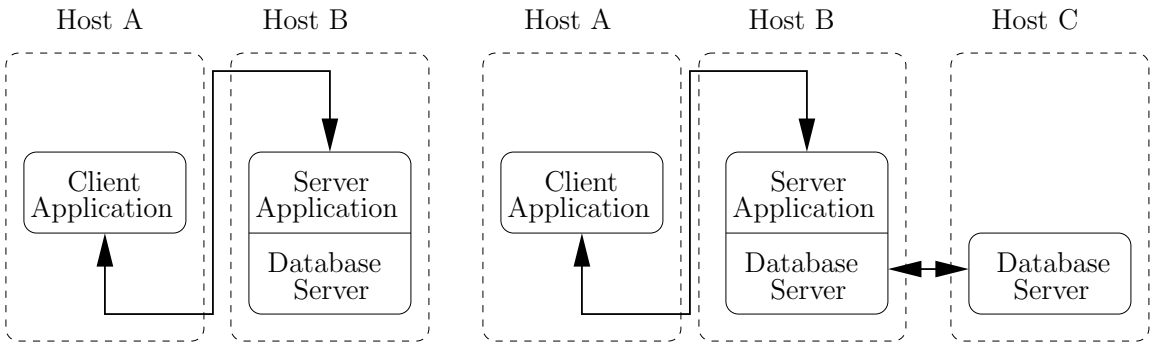
e. One of the most important tasks a server has to perform is security enforcement and authentication. Depending upon the nature of a client-server interface, a server could have to ask a client to properly identify itself.

f. Two-tiered vs three-tiered servers (Fig.12)

(a) The database servers on some servers may have to function as clients of other database servers in order to fulfill client service requests. These type of servers are called three-tiered servers. Otherwise the servers are called two-tiered servers.

(b) Three-tiered servers are increasing in numbers as distributed computing becomes more popular.

(c) The second database server may not necessarily reside on a different host, although in most cases it does.



(a) Two-Tiered (b) Three-Tiered

Figure 12: Two-tiered and three-tiered servers

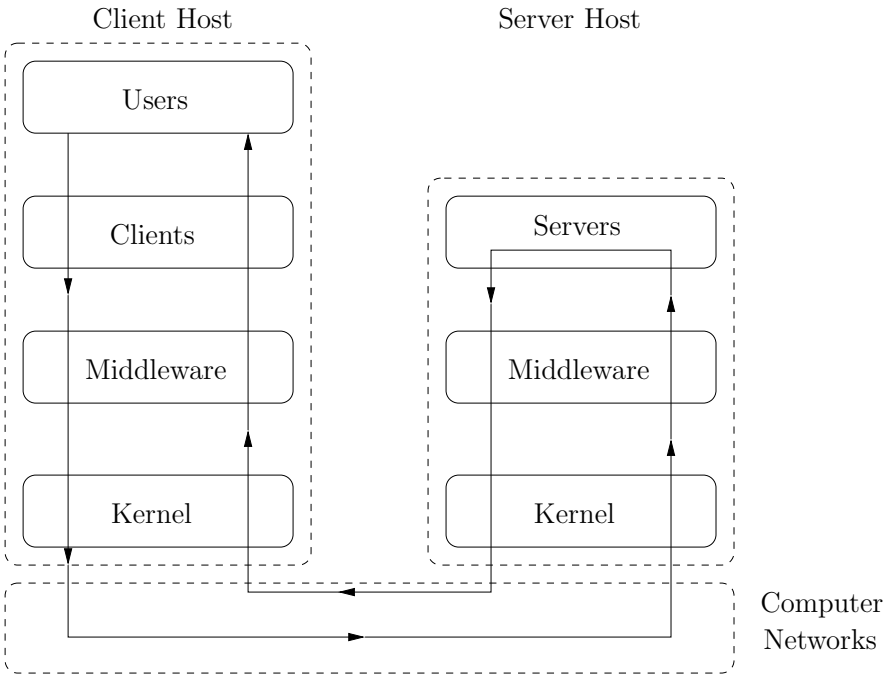


Figure 13: Client-server model from system's point of view

(5) System's point of view (Fig.13)

a. From system's point of view a client-server application is just another pair of application processes (most likely running on different hosts) that are part of the *application layer* in ISO's OSI (open systems interconnection) model.

b. The system provides basic communication transport services (TCP and UDP) as well as datalink services (IP) to all kinds of applications, including client-server model based applications.

3. Format of client requests and their processing

(1) Format of client requests

a. The format depends on the client-server interface (i.e. the protocols). Examples:

(a) It can be as simple as command-line parameters.

(b) It can be very complex. This is particularly for GUI-based and web-based clients. All information in a client's web page has to be translated into a sequence of ASCII characters (or binary characters) as a client request.

b. Interface languages interpreters

(a) To deal with complex client requests, flexible (and powerful) interpreting (also called scripting) languages such as Perl, Javascript, and PHP have been used to parse and extract information from client requests.

(b) Because client requests must follow strict format (as regulated in the client-server interface), they can be thought forming an interface between clients and servers. Perl, Javascript, PHP, TK/TCL, Python, and etc can be regarded as programming languages that can interpret the interfaces.

(2) CGI and CGI programming

a. The HTTP protocol

(a) The protocol specifies the interactions of WEB clients and servers.

(b) As part of the specification, the HTTP protocol specifies the format of client requests and server responses.

b. CGI: common gateway interface

(a) A web server application can be regarded as a gateway to web clients that allow the clients to interact with resources (databases and others) on the server application side.

(b) Example: an on-line membership application web page.

c. CGI programming

(a) CGI programs: they work with HTML pages and HTTP protocols.

(b) CGI programs take as input web client requests. These client requests are generated from complex web pages that may contain forms and other menu items.

(c) CGI programs will decode the client requests data, process them, and send replies if necessary.

(d) CGI programs may call other executable programs residing on the server host to process client requests.

(e) CGI programs are normally started as responses of service requests from clients. They may also be started as responses of a Server Side Include (SSI) command execution.

4. DBMS SPIs

(1) Database languages

a. DDL (data definition languages)

b. DML (data manipulation languages)

c. SQL (structured query language)

(2) Database programming languages

a. High level languages that can interact with database system directly

b. SQL is not a database programming language

c. JDBC

(3) Host programming languages with embedded database APIs

a. Motivations

b. Oracle Pro* C/C++

c. SQLJ

(4) Oracle PL/SQL

a. Motivations

b. Main features

(a) Close ties with SQL and Oracle databases

(b) Ada-like syntax/semantics and modular program structure

i. Block structure programs

ii. Procedures/functions and packages.

iii. Exception handling

(5) Oracle Pro* C/C++ precompiler

a. Oracle Pro* precompilers allow Oracle SQL statements and declarations to be embedded as special blocks into host languages such as C/C++, Ada, Cobol.

b. Principle

(a) Oracle servers provide header files and library files.

(b) The precompiler convert all embedded SQL blocks, type definitions, variable declarations, and any other constructs related to Oracle databases into standard C/C++ function/procedure calls, C/C++ type definitions, C/C++ variable declarations, and other C/C++ constructs.

(c) The converted source programs are pure C/C++ programs that can be compiled directly using standard C/C++ compilers. The resulted object files are then linked using Pro* C/C++ libraries that contain object code for the converted C/C++ functions/procedures.

c. Advantages:

(a) Oracle Pro* C/C++ precompiler enables the combination of C/C++ programming languages and SQL's database access/manipulation capability.

(b) Oracle Pro* C/C++ programs can run in all different platforms support Oracle servers.

(6) SQLJ

a. SQLJ, by its name, allows SQL statements to be embedded into Java programs.

b. SQLJ and PL/SQL complement each other. PL/SQL is more efficient for SQL-intensive applications, while SQLJ is better for portability and for logic-intensive applications.

(7) JDBC

a. JDBC is an API that allows database access in Java.

b. JDBC consists of a collection drivers implemented as Java classes that allow an application to send SQL statement to an Oracle server for execution.

c. Advantages:

(a) JDBC has been supported by major database vendors (Oracle, Sybase, Informix, DB2, and MS SQLServer). JDBC drivers provide a high-level middleware that shields database server dependent aspects of individual database systems.

(b) Java's portability allows Java applications with JDBC to run on different hardware architecture as long as JDBC support is available.

(8) ODBC (Open Database Connectivity)

a. ODBC is a well-accepted standard for applications to connect and access database servers.

(a) ODBC defines an API that applications can use to connect to databases and access/manipulate data in the databases.

(b) The API provided by ODBC can be used by many type of applications written in many host languages.

b. ODBC is well accepted. Most current database vendors support ODBC by providing a ODBC library that contains object code of ODBC API functions.

(9) Oracle call interface (OCI)

a. OCI is an API that allows programmers to create applications that access Oracle database servers through calling functions/procedures in high-level languages such as C/C++, Cobol, and Fortran.

b. OCI consists of a collection of functions and procedures with uniform syntax and semantics across different host languages. OCI functions/procedures are not embedded into host languages (because they are functions/procedures native to the host languages), hence there is no need of any precompilers.

(10) PHP

a. Started in 1994 by *Rasmus Lerdorf*.

(a) PHP: originally stands for *Personal Home Page*.

(b) Current meaning: *PHP Hypertext Processor*

b. PHP is a *web programming language*

(a) PHP code can be easily embedded in HTML files. The execution of this code can produce *dynamic contents*.

(b) Support of interactions with almost every major database systems (to be mentioned shortly).

(11) XML (Extensible Markup Language)

a. Evolved from HTML and SGML (standard generalized markup language), XML is not originally designed to work with database applications.

(a) HTML is a standard text-formatting language. HTML documents contain information as how to format and display the documents.

i. XML is both a text-formatting and processing language. XML documents contain information about how to format the documents, as well as information about the nature (functionality) of documents. The latter information enables XML documents to be processed by recipients.

ii. XML is a language that is understandable by both human readers and computer applications

(b) Because XML documents contains functionality information, they have been widely regarded as a better way for information exchange.

b. Every XML document contains elements, attributes, and sub-elements. As with all markup languages, the markups in XML in are expressed using *tags* within pairs of angle-brackets.

(a) Elements are tagged. XML documents consume additional storage due to use of tags.

(b) An element can have attribute that is specified as part of the open tag.

c. Storage of XML documents

(a) XML documents can be stored directly as XML data (flat) files. They can be retrieved and then processed using XML querying and transforming utilities.

(b) In XML databases, which are databases built using XML as its data model.

(c) XML documents can also be stored in popular relational databases.

d. Querying and transforming XML documents

(a) Querying. XML documents may contain information that can appear in typical relational databases (such the bank enterprise example in DB textbook). Many applications would like to query such XML documents, just querying a relational database.

(b) Transforming. For many other XML documents, there is need to convert from one representation (expressed in one set of XML schemas) to another representation (expressed in another set of XML schemas).

e. Popular XML querying and transforming languages

(a) Xpath. A language for path expressions of XML documents.

i. XML itself does not provide a method for locating specific piece of information in an XML document.

ii. Xpath is a language that can be used to efficiently locate specific piece in an XML document.

iii. Xpath itself is the foundation of most other higher level and more powerful XML querying and transforming languages.

(b) XSLT (XSL Transformations. XSL stands for XML Stylesheet Language).

(c) XQuery. A language proposed by W3C as a standard XML query language.

i. Based on Quilt (an early XML query language), XQL and XML-QL

ii. The FLWR (*for, let, where*, and *return* expressions

f. XML APIs – programming interfaces that allow access/manipulation of XML documents from many programming languages

(a) DOM (document object model), is an API based object-oriented model

i. Treating XML documents as trees – consistent with the original XML structure. Access/manipulation of an XML document is through navigation of the corresponding XML tree.

ii. Available in many popular PL in the form of DOM libraries

(b) SAX (simple API for XML), an event model based XML API

i. XML parsers: software that parse XML documents to check their syntax conformance.

ii. Designed to provide an interface between parsers and XML applications. Built on the notions of event handlers.

g. XML and interfaces of Internet applications

(a) XML supersedes HTML as a document formatting language.

(b) The ability of specifying document processing information, being transformed from one XML document to other kind of documents, being used as both a database storage format and storage language, and being queried subsequently, also makes XML an ideal language for Internet applications and as interfaces of such applications.

5. Perl vs PHP

(1) Commonality

a. Both are interpretted languages, have been heavily used for scripting.

b. Perl was developed earlier. PHP follows the syntax of Perl and C.

c. Good portability.

(2) Uniqueness

a. Unique features of Perl

(a) Earlier than PHP. Well known and has larger usage base.

b. Unique features of PHP

(a) Web programming language:

· Built-in native support of HTTP methods;

· Close interactions with HTML language.

(b) Database integration:

· Built-in native connection facility with most of popular database systems such as MySQL, Oracle, Informix, Sybase, PostgreSQL, DBS2, Microsoft SQL Server, mSQL.

· Support of ODBC (open database connectivity).

(c) Object-oriented: Support of most object-oriented features of modern OO programming languages.

(d) Built-in libraries: has built-in functions that pertain to many useful Web-related functions.

· Document manipulation functions (such as postscript, PDF, GIF file generations).

· Cookies management.

· Networking services, email services.