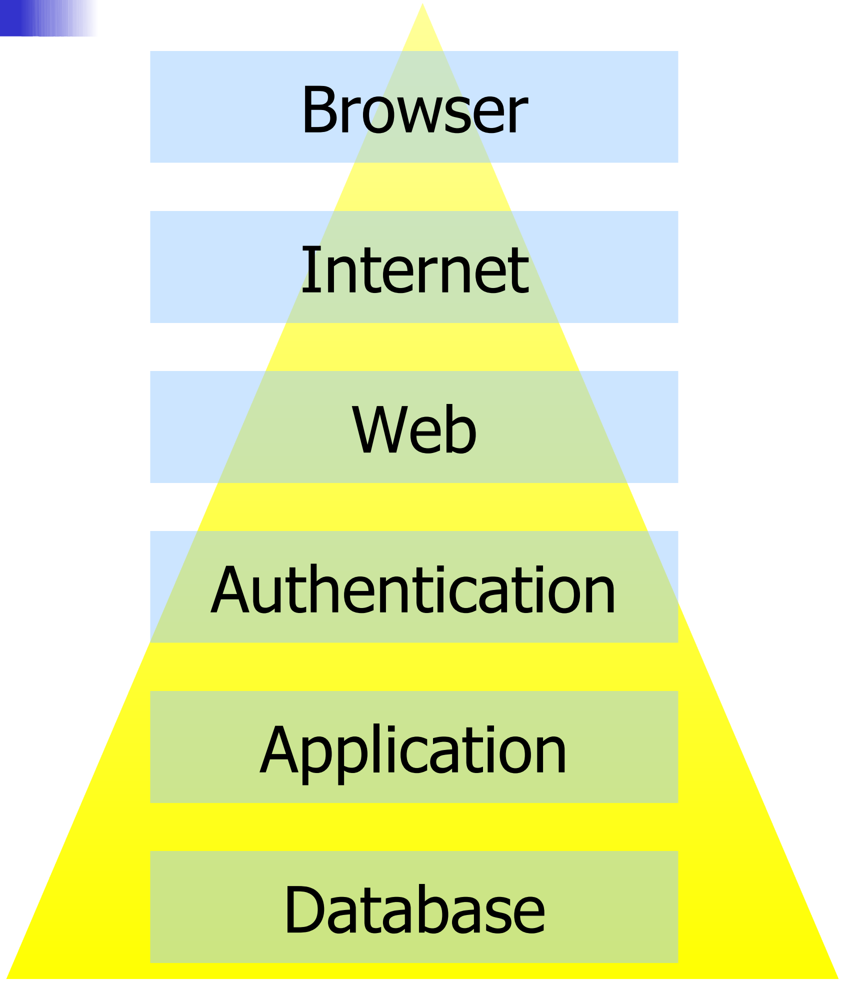
Web-based Architectures

Fulvio Corno, Dario Bonino

Outline

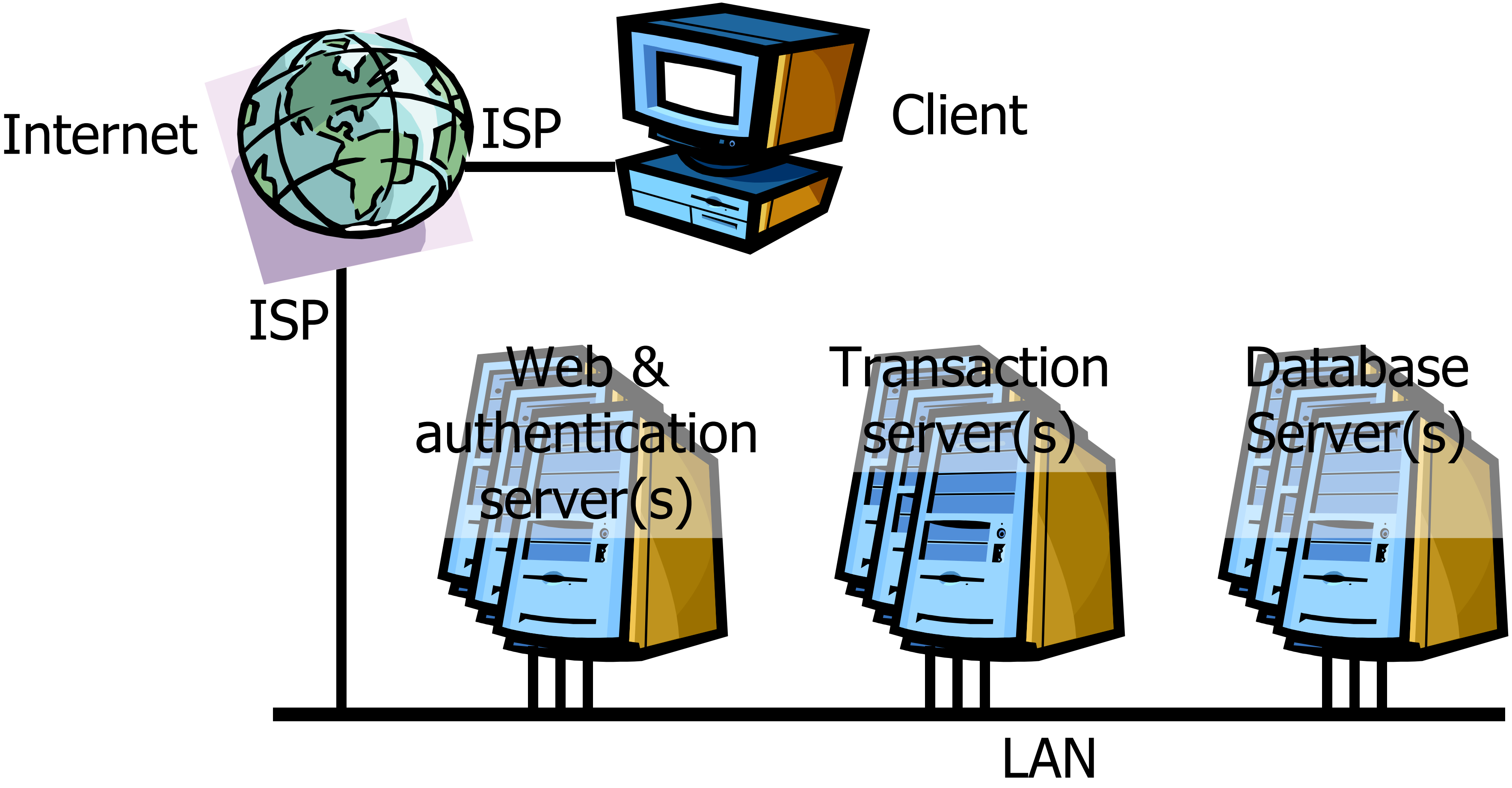
- Reference Architecture
- Web server
- Application server
- Database server
- Networks, proxy, cache
- Load balancer
- Enterprise frameworks

N-Tier architectures



- Each level/tier has a well defined role
- One or more servers implement each tier/layer
- More servers can share the same hardware or can run on dedicated devices
- Communication between tiers/levels is achieved through the network

General Architecture



Components

- One or more connections to the Internet by means of an Internet Service Provider (ISP).
- One or more servers implementing each tier/level of the architecture.
- One or more physical networks for interconnecting the servers.
- One or more network devices (router, firewall, switch) which implement communication and security policies.

What is a *server*?

Two senses:

- A process that runs on a host that relays information to a client upon the client sending it a request.
- A host computer on a network that holds information (eg, Web sites) and responds to requests for information

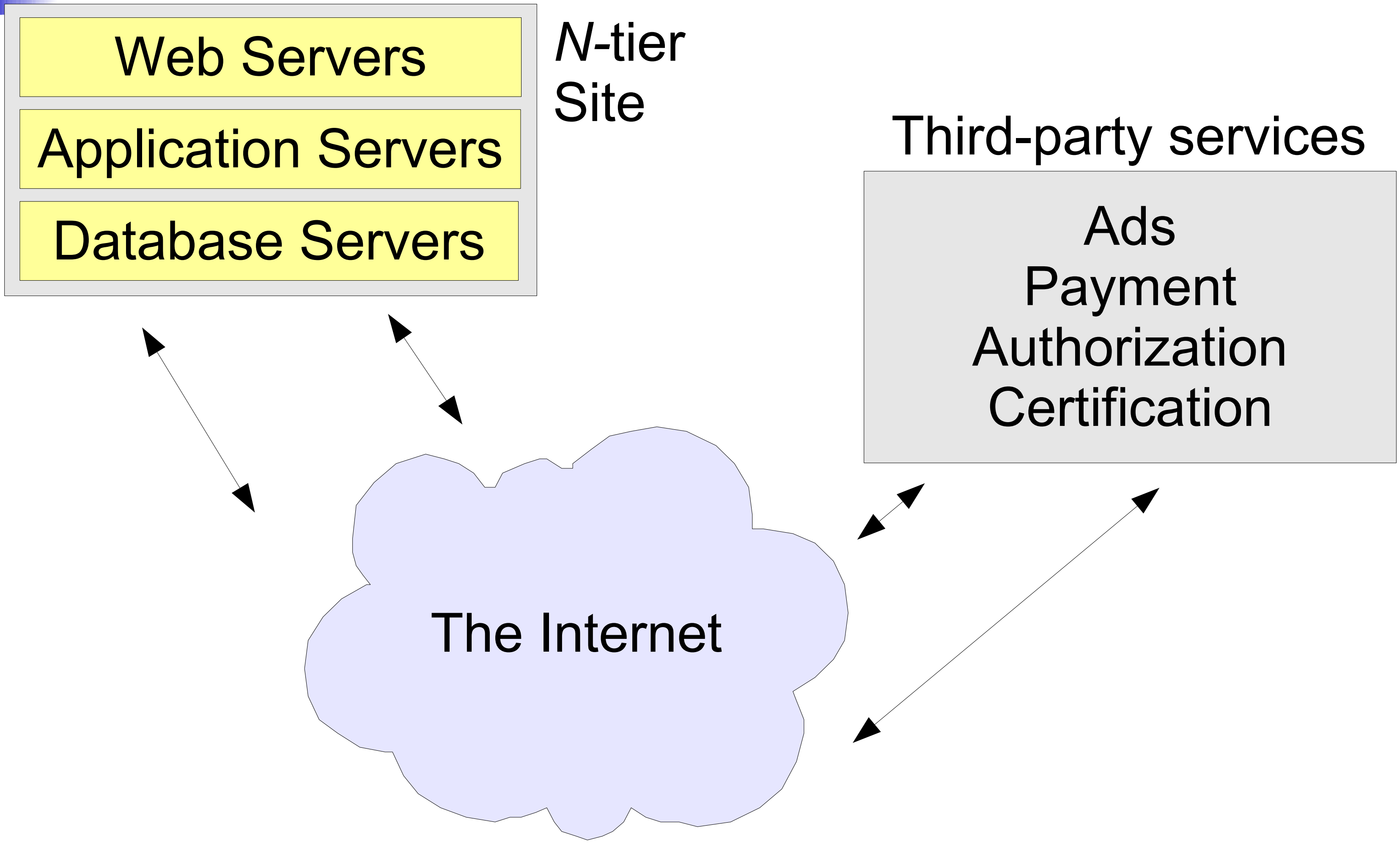
In our reference architecture the first sense is better (many servers can run on the same host computer).

What is a *server* ?

The offered functionalities depend on:

- The kind of server software (daemon / service)
- The network stack on the host
- The operating system
- The hardware
  - CPU
  - Storage & Memory (I/O)
  - Communication capabilities (Network)

e-Business Architecture



Web Servers

Application Servers

Database Servers

N-tier
Site

Third-party services

Ads
Payment
Authorization
Certification

The Internet

Clarifications

- Architectures can be very different, depending on the site/application requirements.
- Usually some services are delegated to third-party, e.g. payments, security, advertisement. These services belong to a different e-Business framework.

Outline

- Reference Architecture
- Web server
- Application server
- Database server
- Networks, proxy, cache
- Load balancer
- Enterprise frameworks

Web server

- Manages the HTTP protocol (handles requests and provides responses)
  - Receives client requests
  - Reads static pages from the filesystem
  - Activates the application server for dynamic pages (server-side)
  - Provides an HTML file back to the client
- One HTTP connection for each request
- Multi-process, Multi-threaded or Process pool

Web Server (2)

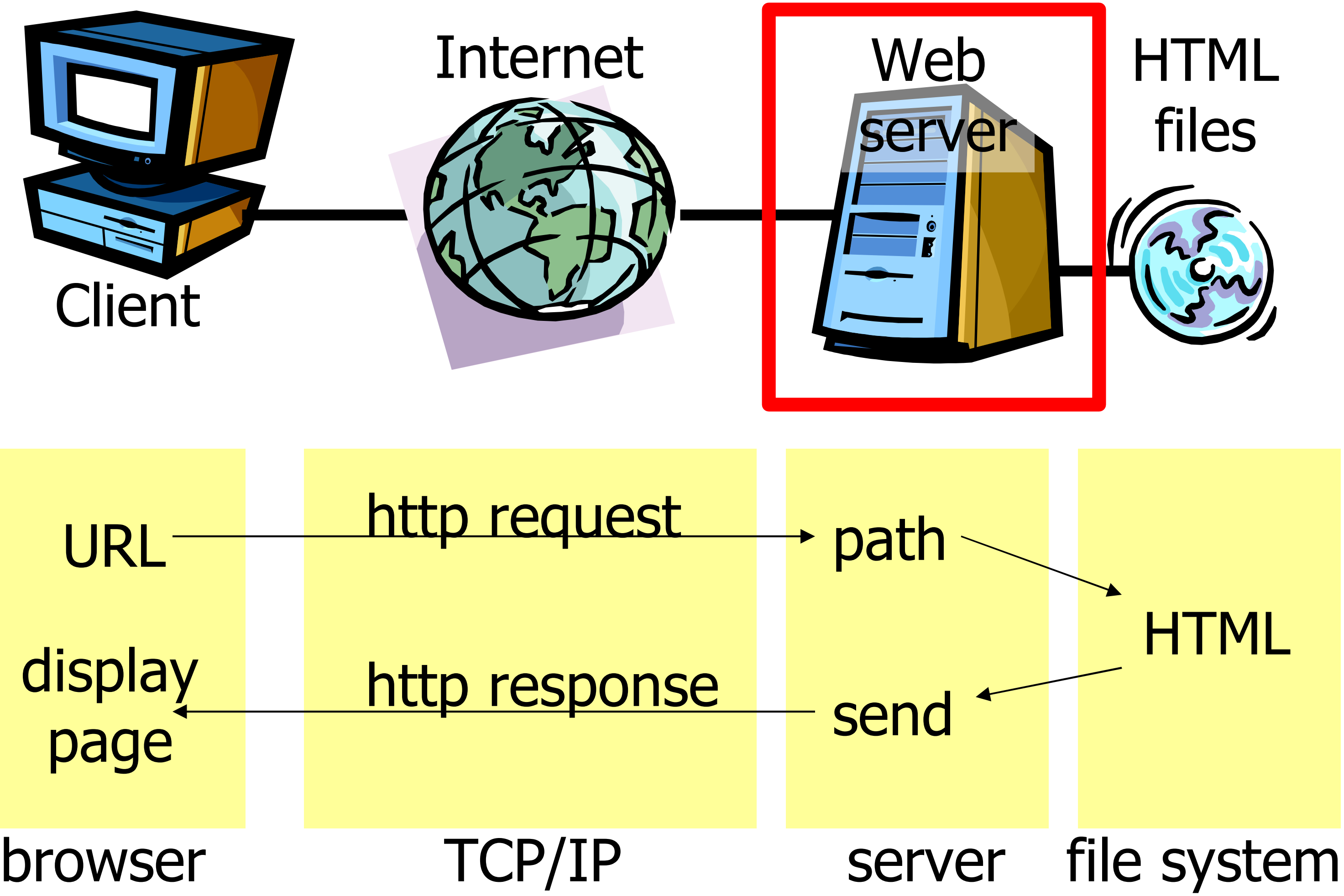
Multi-threaded

- means that inside each server's process, there are two or more threads, each one able to execute its own task independently from the others
- threads share a main process context, a crashing thread may more easily crash the whole application

Multiprocess

- A pool of processes are used, and reused.

Example



Client

Internet

Web server

HTML files

URL

display page

http request

http response

path

send

HTML

browser

TCP/IP

server

file system

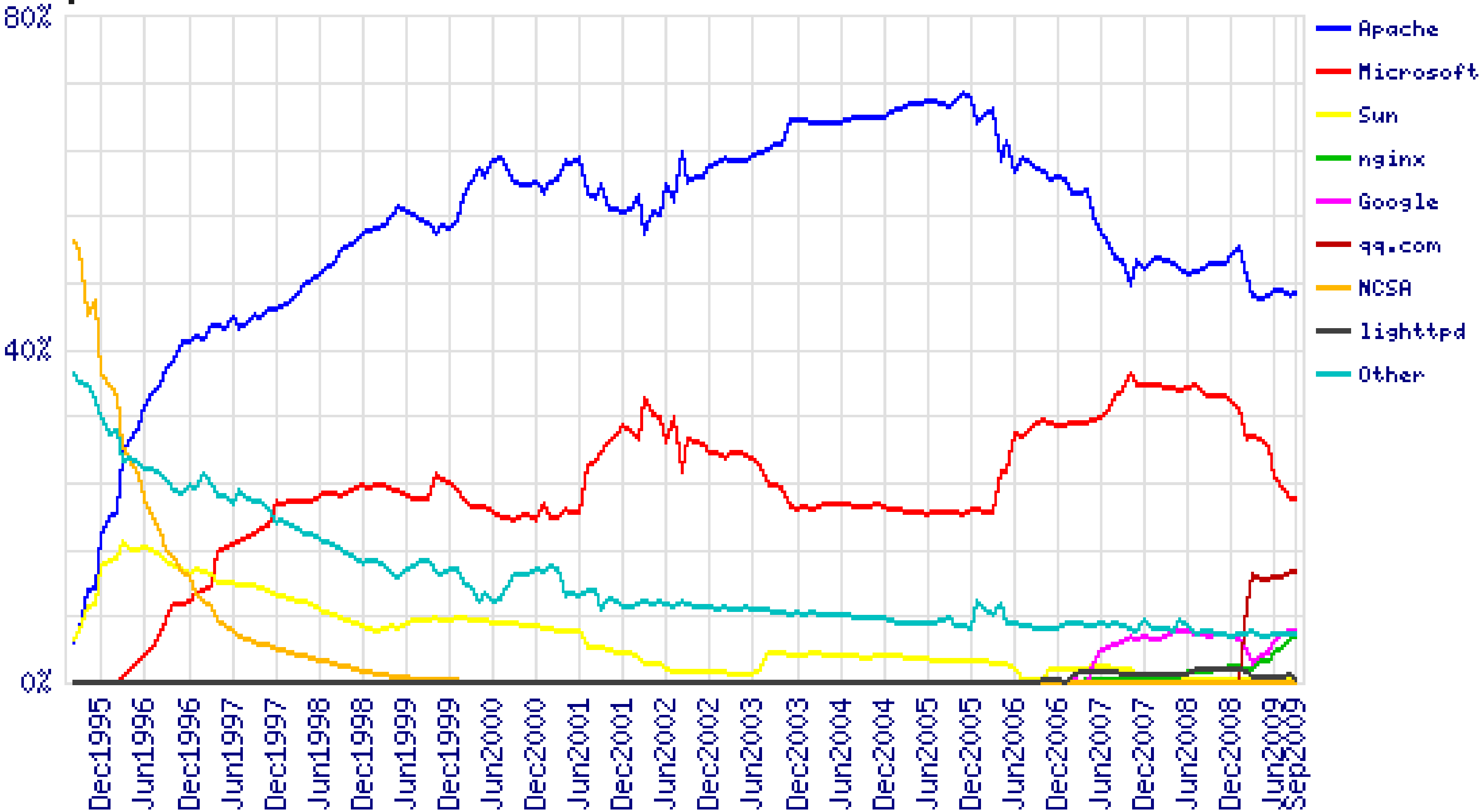
Performance measures

- Latency: time required for providing a 0 byte http page. Includes the server activation time, the request decoding time, the file access time, the transmission time and the time for closing the connection.
  - Unit of measure: http/s or s/http
- Throughput: maximum speed at which infinite-sized pages can be sent.
  - Unit of measure: Bytes (Mbytes)/s
- #Requests / s

Delay time

- T = Latency + ByteHTML / Throughput
- This equation is valid if:
  - The other architecture elements (I/O subsystem, network, ...) are not overloaded
  - The web server has not yet reached its maximum workload
- Example:
  - Latency: 0,1s
  - ByteHTML: 100kBytes
  - Throughput: 800kBytes/s
  - T= 0,1s+ 100KBytes / 800KBytes/s =0,225s

The most adopted web servers



Source: http://news.netcraft.com/

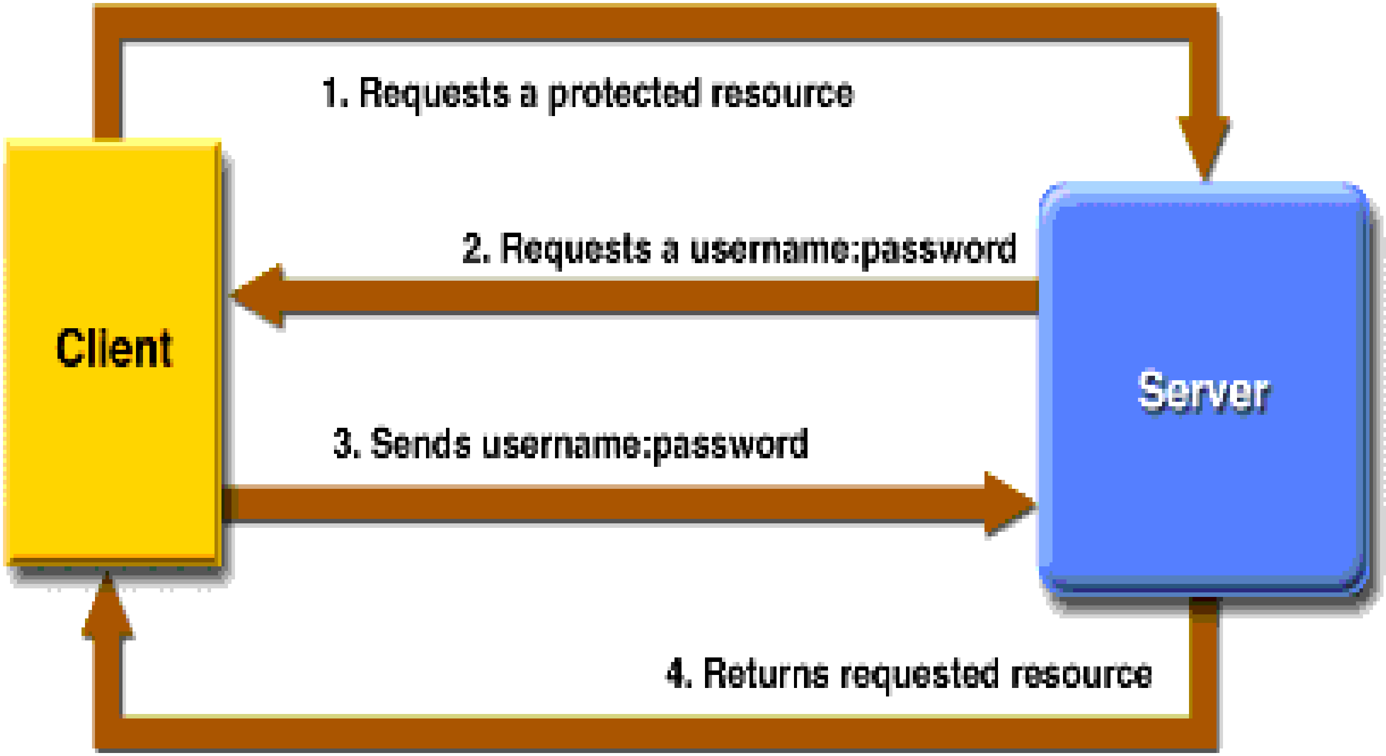
Authentication server

Web Servers are often used to verify the user identity.

There are several authentication levels:

- Session (cookie)
  - Based on username/password
  - Application level
- Http authentication
  - Web server
- Digital signature
  - Web server extensions

HTTP Authentication


1. Requests a protected resource
2. Requests a username:password
Client
Server
3. Sends username:password
4. Returns requested resource

Secure Web Server

Most Web Servers can handle authentication and encrypted communication for security-sensitive information, such as payment transactions, and corporate log-ons.

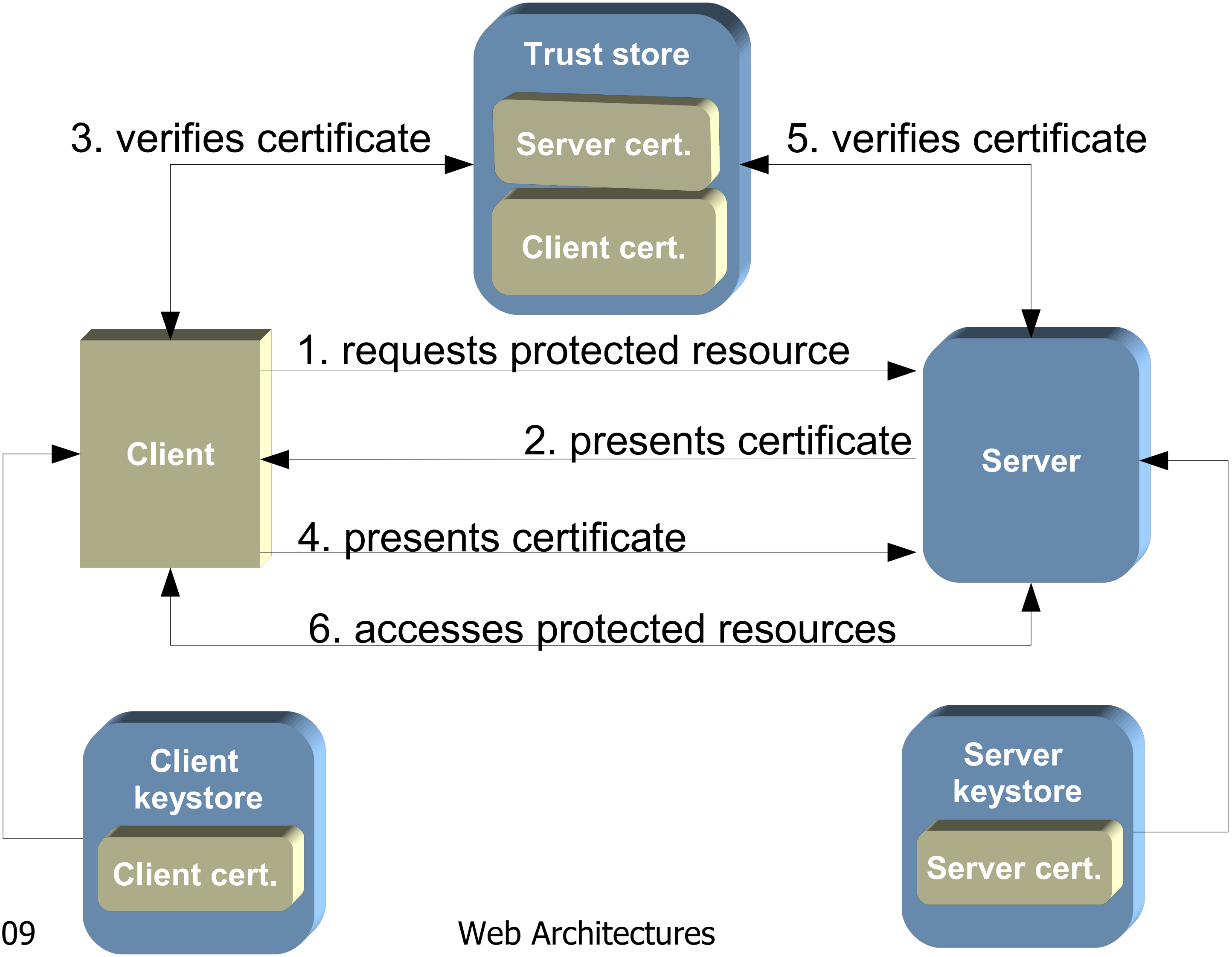
Secure communication has impact on the Sever performances (slower)

Secure Web Server (2)

How does it works?

- To prepare a web-server for accepting secure (https) connections the administrator must create a public key certificate for the web-server.
- This certificate must be signed by a certificate authority, who certifies that the certificate holder is who he says he is
- Certificates are distributed to the site users (they load them in their browser)

Secure Web Server (3)



Trust store

Server cert.

Client cert.

3. verifies certificate

5. verifies certificate

Client

Server

1. requests protected resource

2. presents certificate

4. presents certificate

6. accesses protected resources

Client keystore

Client cert.

Server keystore

Server cert.

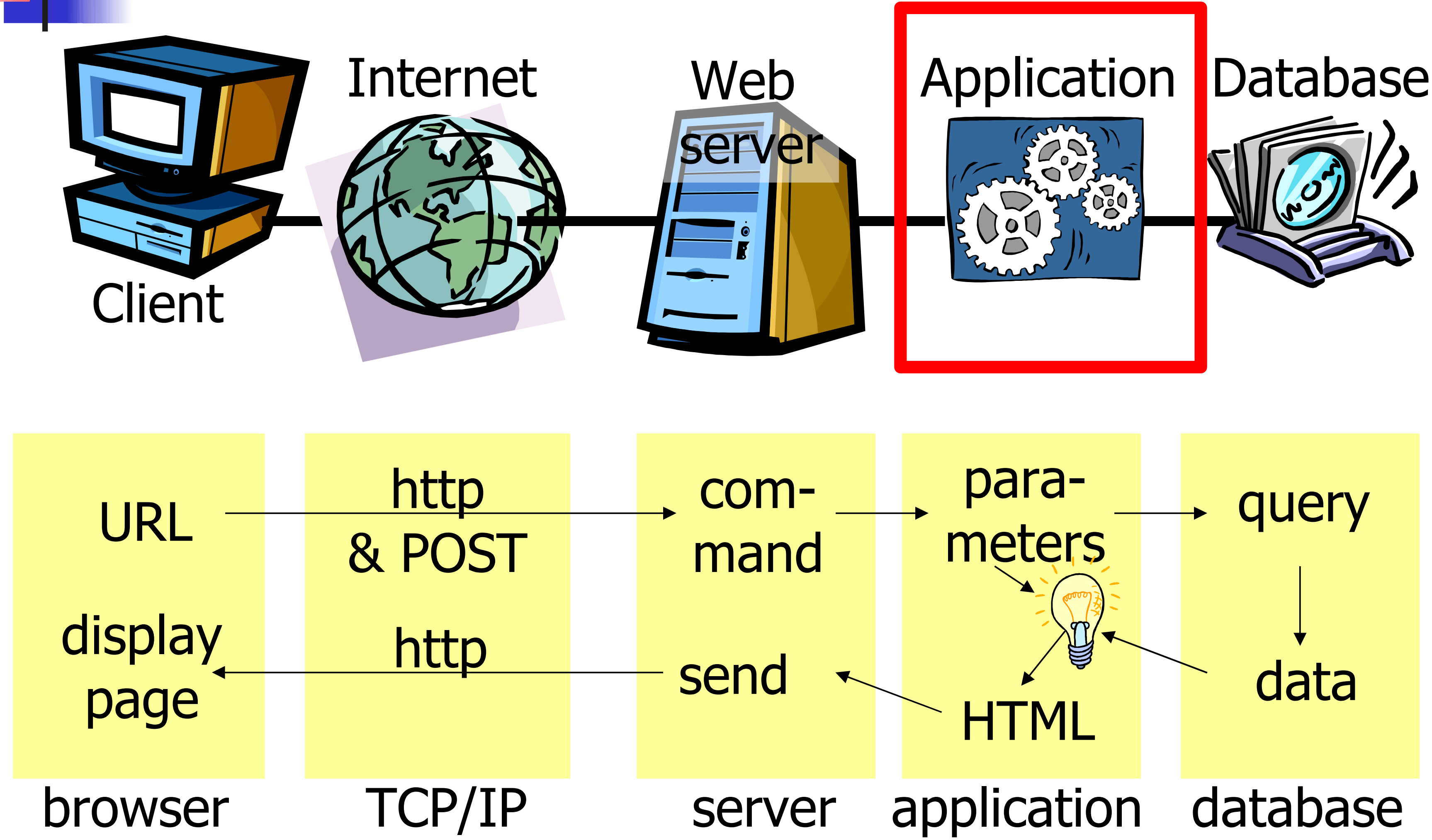
Outline

- Reference Architecture
- Web server
- Application server
- Database server
- Networks, proxy, cache
- Load balancer
- Enterprise frameworks

Application server

- Dynamic page generation
- Manages the site business logic
- It's the middle tier between the client browser and the data residing on a database
- Implements the session mechanisms
- Different technologies and architectures are available

Example



Client — Internet — Web server — Application — Database

browser	TCP/IP	server	application	database
URL	http & POST	command	parameters	query
display page	http	send	HTML	data

Dynamic page generation

- Elaboration of user-provided data / requests
- Page generation from content repositories (database)
- Provision of time-dependent information

User provided data

- Users can send data to a given site for:
  - Logging in
  - Performing queries
  - Reserving flights or hotels
  - …
- Data is gathered by means of FORMs
  - XHTML FORMs
    - Several input fields
    - A submit button

XHTML FORMs

- In XHTML
  - <form method=”GET” action=”search”>
    ....
    </form>
  - <form method=”POST” action=”search”>
    ....
    </form>
- GET transfers data in the URL
- POST transfers data in the HTTP message body

XHTML FORMs (2)

- GET
  - Data is encoded in the URL (URL-encoding)
  - GET http://www.mysearch.com/search?query=coffee
- POST
  - Data is encoded in the HTTP message
  - POST http://www.mysearch.com/search
  - Message body: query=coffee
- In both cases /search identifies the application to be executed by the Application Server

Technologies

The application server can manage:

- Standalone applications (run by the application server)
  - CGI
  - Fast CGI
- Server-side scripting (interpreted by the application server)
  - ASP (Active Sever Pages)
  - PHP (PHP: Hypertext Processor)
  - JSP (Java Server Pages)

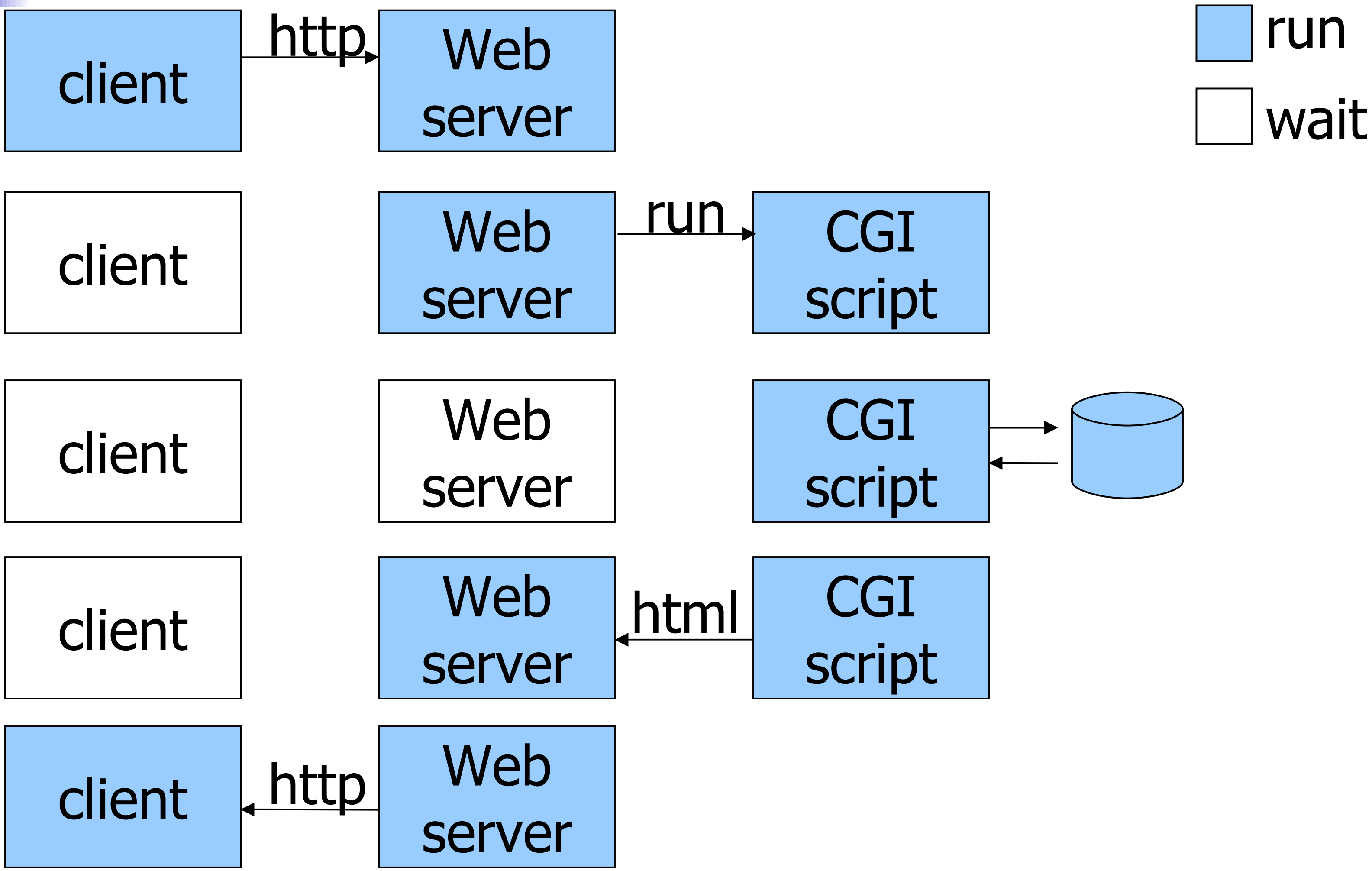
Technologies (2)

- Enterprise frameworks
  - J2EE
    - Servlets
    - EJBs (Enterprise Java Beans)
    - Web Services
  - .NET
    - ASP .NET
    - Web Services

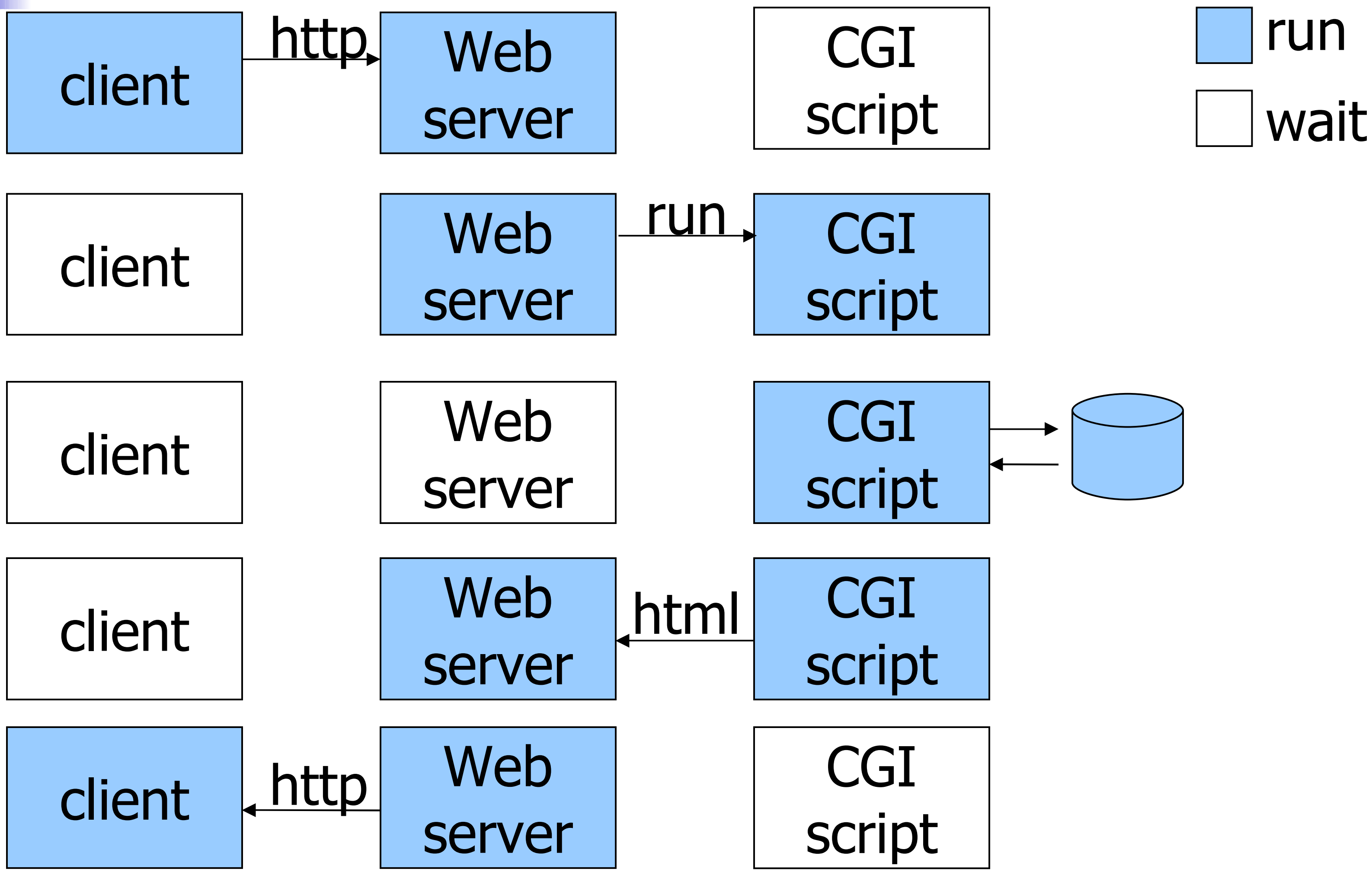
CGI / Fast CGI

- CGI stands for “Common Gateway Interface”
  - CGI is a standard protocol for interfacing external application software with an information server, commonly a web server. (wikipedia)
  - For every request a new application process is spawned
- Fast CGI
  - Speeds up the CGI by using a multi-threaded persistent process
- the process is executed by the host Operating System (OS)

CGI


client
http
Web server
client
Web server
run
CGI script
client
Web server
CGI script
client
Web server
html
CGI script
client
http
Web server
run
wait

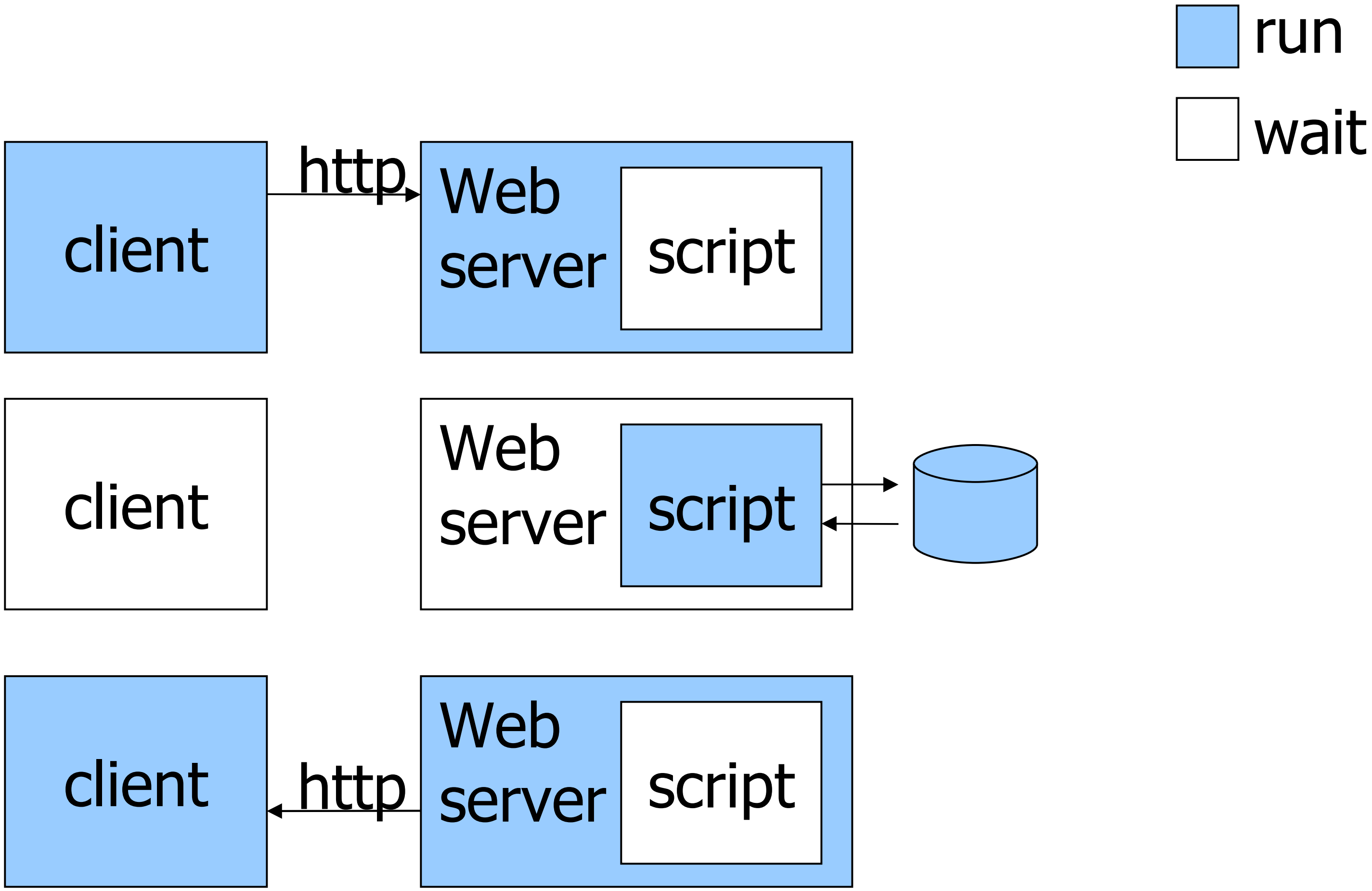
FastCGI


client
http
Web server
CGI script
run
wait
client
Web server
run
CGI script
client
Web server
CGI script
client
Web server
html
CGI script
client
http
Web server
CGI script

Sever-side scripting

- Scripts: programs written in different languages (Visual Basic, Java, PHP)
- They are interpreted by the Web Server directly or through extension modules
- Faster than CGI as
  - Interpreters are loaded together with the web server
- Pages are actually mixes of
  - XHTML
  - Instructions to be interpreted

Server scripting (ASP, JSP, PHP, …)

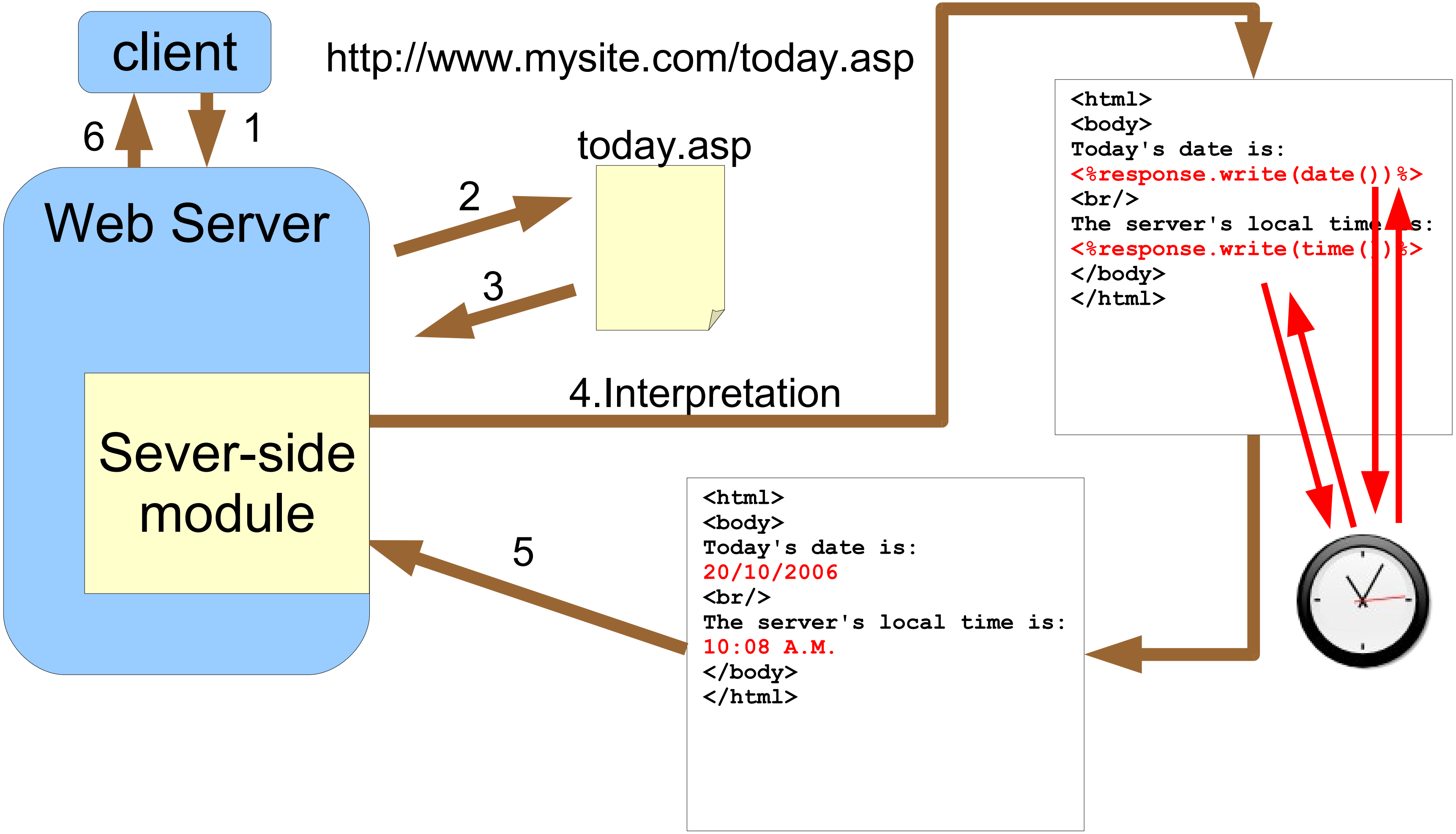


ASP example

```
<html>
<body>
Today's date is: <
%response.write(date())%>
<br/>
The server's local time
is: <
%response.write(time())%>
</body>
</html>
```

```
Today's date is:
10/20/2006.
The server's local time
is: 10:08:30 AM.
```

ASP Example (2)



```
<html>
<body>
Today's date is:
<%response.write(date())%>
<br/>
The server's local time is:
<%response.write(time())%>
</body>
</html>
```

```
<html>
<body>
Today's date is:
20/10/2006
<br/>
The server's local time is:
10:08 A.M.
</body>
</html>
```

PHP example

```
<html>
<body>
Today's date is: <?php
echo date(“d/m/Y”);?>
<br/>
The server's local time
is: <?php echo
date(“h:i:s”);?>
</body>
</html>
```

Today's date is: 10/20/2006.

The server's local time is: 10:08:30 AM.

JSP Example

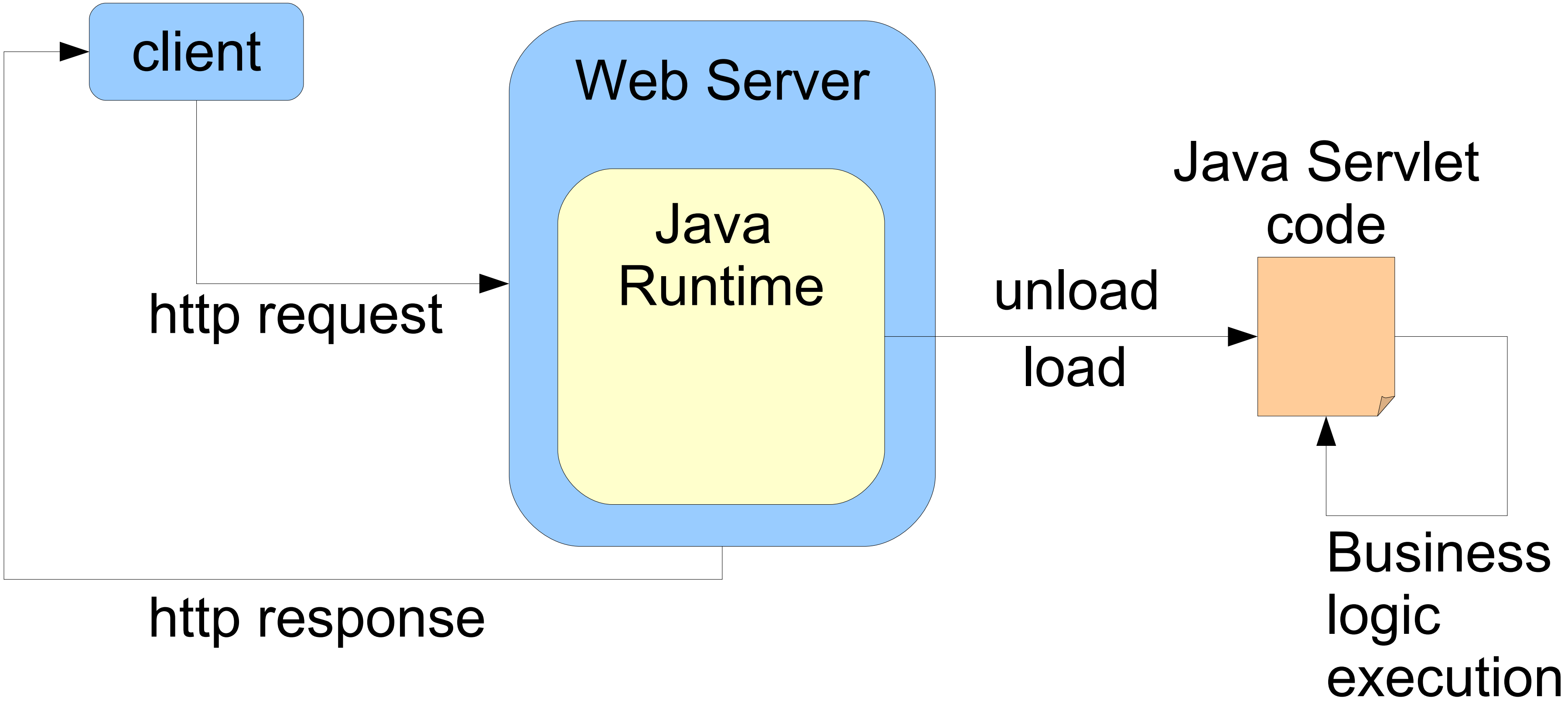
```
<%@page import="java.util.*, java.text.*" %>
<html>
<body>
Today's date is:
<%
Date today = new Date();
SimpleDateFormat formatter = new
SimpleDateFormat(“dd/MM/yyyy”);
out.println(formatter.format(today)); %>
<br/>
The server's local time is:
<% formatter.applyPattern(“hh:mm:ss a”);
out.println(formatter.format(today));%>
</body>
</html>
```

Today's date is: 10/20/2006.
The server's local time is:
10:08:30 AM.

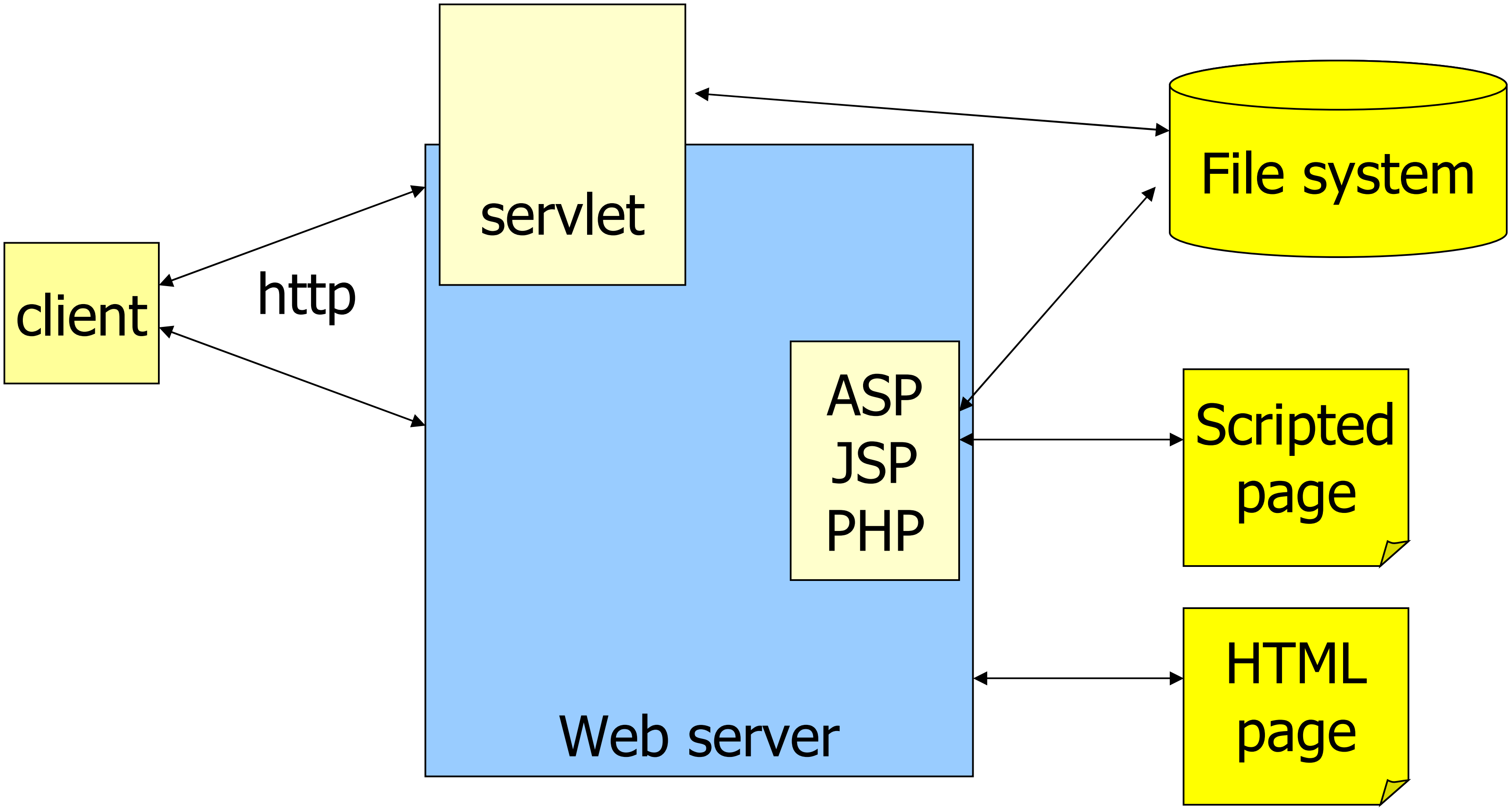
Enterprise frameworks

- Quick glance on servlets… more data afterwards
- The application server is a “servlet container”
- Servlet are Java Based (so a Java Virtual Machine is needed)

Java Servlets



Summary


servlet
File system
client
http
ASP
JSP
PHP
Scripted page
HTML page
Web server

Web Sessions

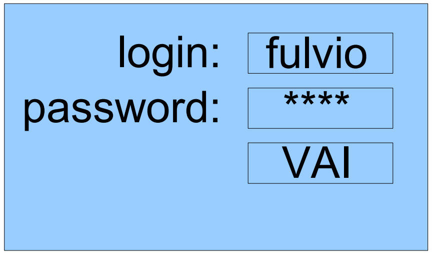
- The HTTP protocol is stateless (without memory)
  - Every request is independent from the previous one(s), and cannot “remember” the previous path of a given user
  - Input variables in each page may only depend on the FORM in the immediately preceding page
- Users, on the other hand, need to feel the impression of a “continuous” navigation, where a choice made in a page is “remembered” for all successive pages.

Requirement

- We must build an **abstraction** of a “navigation session”
- Session = sequence of pages, visited on the same site by the same user with the same browser, in the same navigation stance, having a logic sequentiality

The problem: example

- The user enters username & password on the first page he visits
- The next page will be a dynamic page “validating” the login
- All successive pages should know the user name...



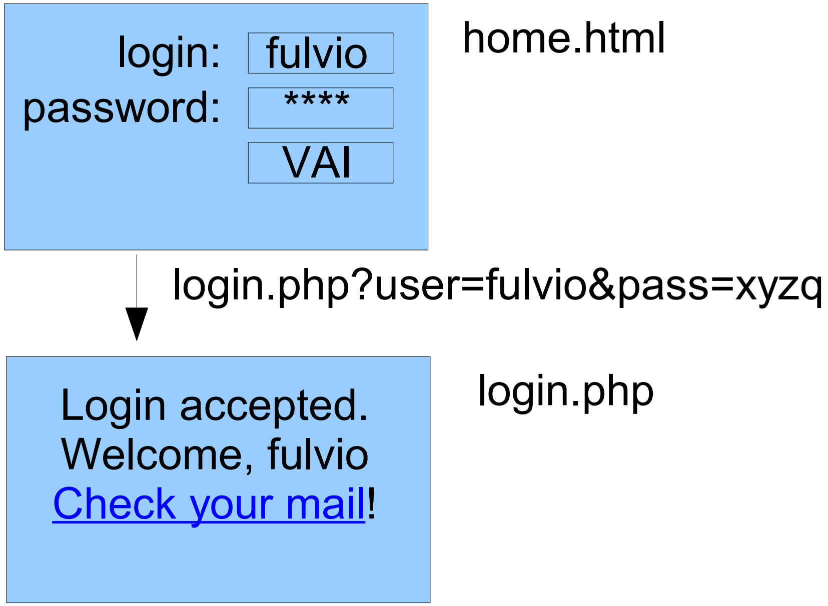
login: fulvio

password: ****

home.html

The problem: example

- The user enters username & password on the first page he visits
- The next page will be a dynamic page “validating” the login
- All successive pages should know the user name…



home.html

login: fulvio

password: ****

VAI

login.php?user=fulvio&pass=xyzq

login.php

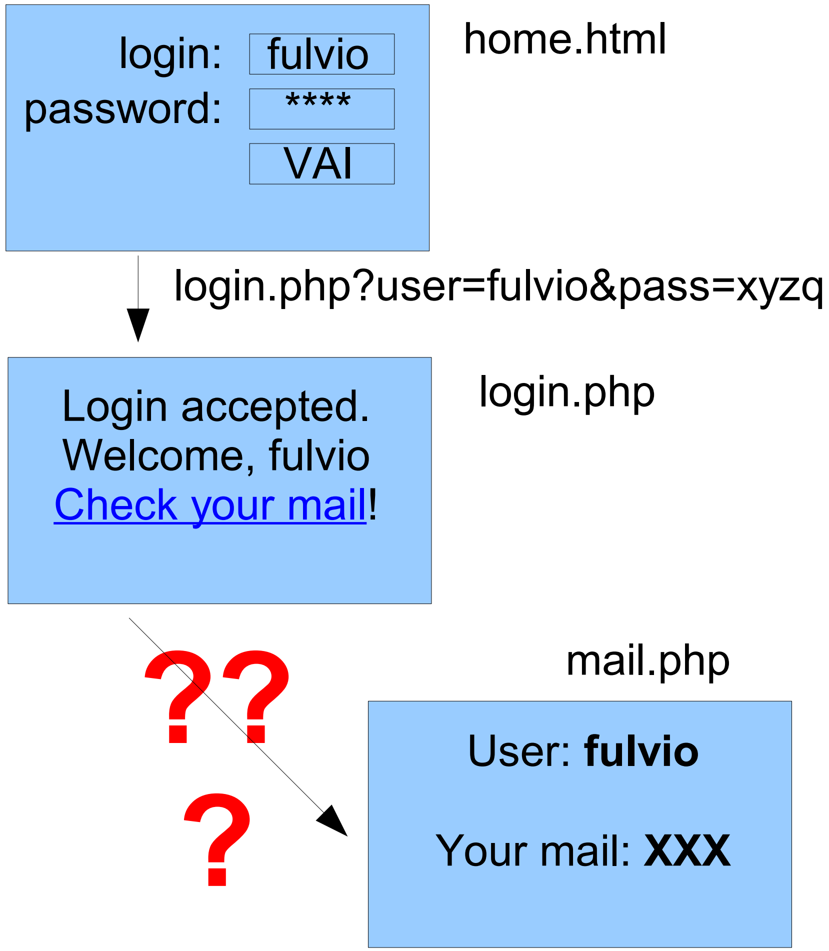
Login accepted.

Welcome, fulvio

Check your mail!

The problem: example

- The user enters username & password on the first page he visits
- The next page will be a dynamic page "validating" the login
- All successive pages should know the user name...



home.html

login: fulvio

password: ****

VAI

login.php?user=fulvio&pass=xyzq

login.php

Login accepted.
Welcome, fulvio
Check your mail!

???

mail.php

User: **fulvio**

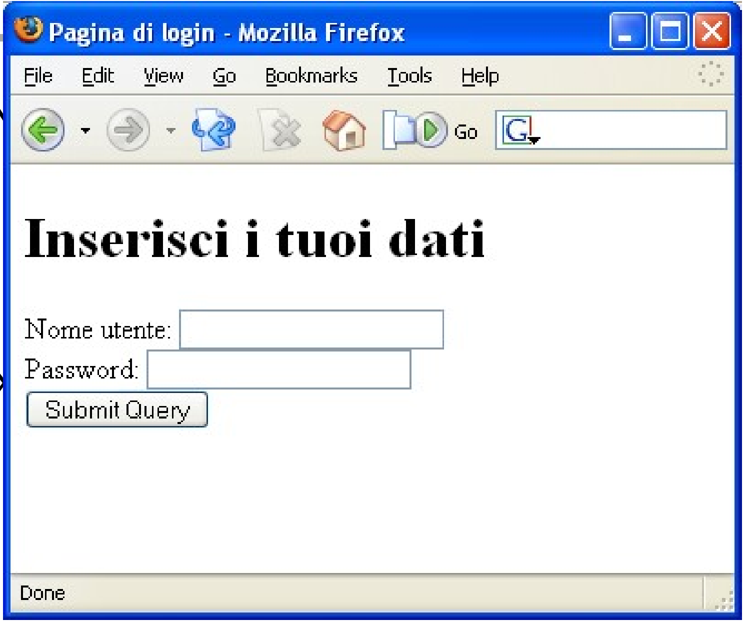
Your mail: **XXX**

Example: home.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN
"http://www.w3.org/TR/html4/strict.dtd">
<html>
  <head>
    <title>Pagina di login</title>
    <meta http-equiv="content-type" content="tex
  </head>
  <body>

     <h1>Insert your data</h1>
     <form method="GET" action="login.php">
       Username: <input type="text" name="user" /> <br />
       Password: <input type="password" name="pass" /> <br />
       <input type="submit" />

     </form>
  </body>
</html>
```



Pagina di login - Mozilla Firefox

Inserisci i tuoi dati

Nome utente:

Password:

Submit Query

Done

Possible solutions

- URL rewriting
- Hidden FORM variables
- Cookies
- Cookie-based session management

URL rewriting

- We may modify the text of *all links* in the pages, to pass every time, with the GET method, all desired variables.
  - <a href="mail.php?user=
    <?php echo "\"" . $_GET["user"] . "\"" ?>
    >
- We must modify *each and every* link with this strategy

Hidden variables

- We may add some hidden variables into a FORM
  - <input type="hidden" name="user" value="fulvio" />
- Invisible to the user, cannot be modified by the user
- Accessible to the server page ($_REQUEST["user"]) and can be modified by the server (value="xxxx")

Hidden variables: drawbacks

- We must repeat *all* variables in *all* pages
- Only FORM-based navigation
  - No <a href="..."> links!

Cookies

- Cookies are "small" parcels of information that the server may store on the browser
  - In the http response, the server adds a "set cookie" command
  - The browser promises to send back all cookies to the server at each successive http request
- A cookie is a 4-tuple composed of
  - Name
  - Value
  - Expiration (date/time)
  - Domain

Cookie-based solution

- When login.php validates the password, it sets a cookie on the browser
- The page mail.php may read the cookie that will be received by the browser at the next request

Cookies in PHP

- Writing: setcookie("name", "value")
  - *bool setcookie (string name [, string value [, int expire [, string path [, string domain [, int secure]]]]])*
  - Must be called **before any** HTML contents (including spaces!)
- Reading: $_COOKIE["name"]
- Checking: if(isset($_COOKIE["name"])) ...
- Deleting: setcookie("name", "")

Cookies: pros

- Only one set-cookie is needed, at the first page receiving the information
- Many cookies may be set, one for each variable
- No need to modify links nor FORMs

Cookies: cons

- Impossible to store “large” information
- Storing a lot of cookies is “impolite”
- Trusting data coming from the browser may be unsafe
- The browser might (legitimately) refuse to send the cookies back

Session management

- We may store many different variables by using *one only cookie*
- The cookie is used to store a “magic number”, different for each new session
  - name="session" value="123456"
- The server stores (somewhere) all session variables associated to that number
  - Session=123456 ⇒ { "user" = "fulvio", "language" = "IT" }

Session variables

- A set of (name, value) couples
- May be stored
  - In the web server memory
  - In a file private to the web server (ex. session.123456.vars)
  - In a database table (Session, Name, Value)
- On every new page, the received cookie will “unlock” the associated variables, and make them available to the server page.

Sessions in PHP (1/3)

- Session activation: session_start()
  - No parameter. Must be called *before any othe action*
  - Creates a new session, or reconnects to the existing one
  - Creates a cookie with a name like PHPSESSID123456

Sessions in PHP (2/3)

- Session variables usage
  - Special vector $_SESSION will store all variables associated to the session
  - $_SESSION["user"] = "fulvio"
  - echo $_SESSION["user"]
  - if (isset($_SESSION["user"])) ...

Sessions in PHP (3/3)

- Session closing:
  - Expires automatically after 20-30 minutes
  - Expires automatically when the browser is closes
  - May be “reset” with session_destroy()

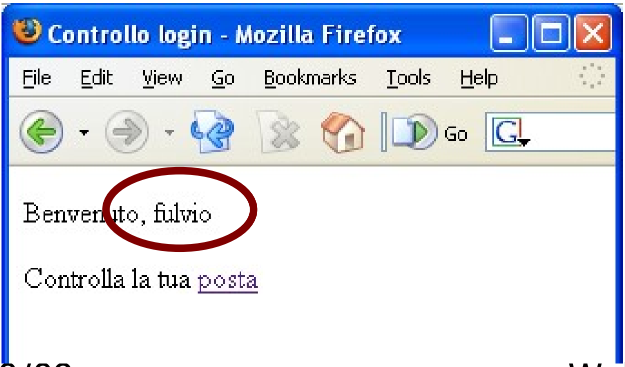
Example: login.php

```
<?php

  session_start() ;

  $user = $_REQUEST["user"] ;
  $pass = $_REQUEST["pass"] ;

  if($user == "fulvio" && $pass == "xyz")
  {
      // login OK
      $_SESSION["user"] = $user ;
      $login = 1 ;
  }
  else
  {
      //
      $l
  }
?>
```



Controllo login - Mozilla Firefox

Benvenuto, fulvio

Controlla la tua posta

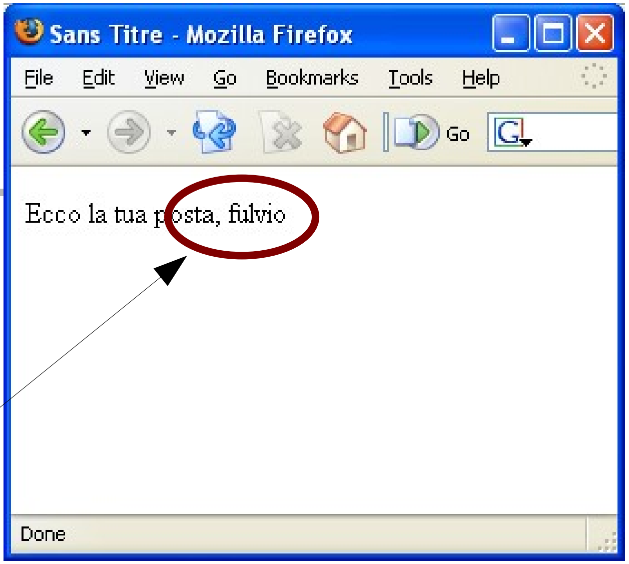
```
<html>
<head>
<title>Check login</title>
</head>
<body>
<?php

if($login == 1)
{
  echo "<p>Welcome, $user</p>\n";
  echo "<p>Check your <a
href=\"mail.php\">mail</a></p>\n" ;
}
else
{
  echo "<p>ERROR: invalid login, <a
href=\"home.html\">retry</a></p>\n" ;
}
//echo "<p>Password: $pass</p>\n" ;
?>
</body>
</html>
```

Example: mail.php

```
<?php
session_start() ;
?>

<html>
<head>
<title>La tua posta</title>
</head>
<body>
<?
   if(isset($_SESSION["user"]))
   {
         $user = $_SESSION["user"] ;
         echo "<p>Ecco la tua posta, $user</p>\n" ;
         /*  stampa la posta... */
   }
   else
   {
         echo "<p>ERRORE: devi prima fare il <a
href=\"home.html\">login</a></p>\n" ;
   }
?>
</body>
</html>
```


Sans Titre - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
Go

Ecco la tua posta, fulvio

Outline

- Reference architecture
- Web server
- Application server
- Database server
- Networks, proxy, cache
- Load balancer
- Enterprise frameworks

Database server

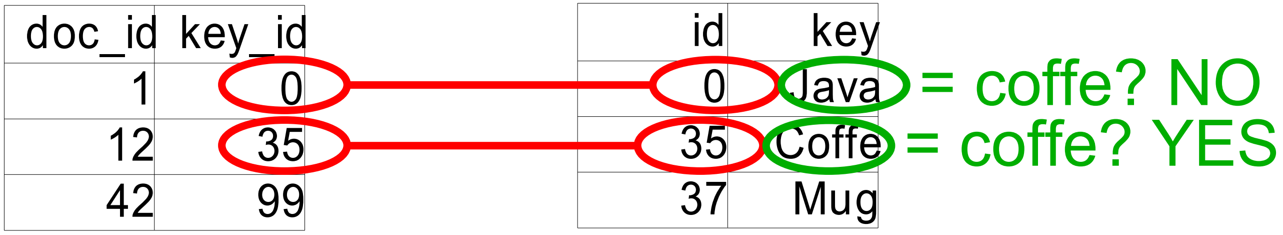
- Stores the data on which the application server works.
- Executes the queries issued by the application server:
  - Updates the stored data
  - Inserts new data
  - Provides back query results
- The most frequent/complex queries can be implemented internally as stored procedures (pre-compiled queries with parameters)

Database server

- Queries are almost always in SQL
  - SELECT * FROM table;
  - ….
- Often adopts the relational database model
  - Other models can be used
    - Object model
    - Triple model
- The most advanced/complete solutions are called Transaction servers

Example

- The “/search” application converts the client-provided parameters “query=coffee” into a database query
  - **SELECT doc_id FROM key_doc_index, keywords WHERE key_doc_index.key_id = keywords.id AND keywords.key = “coffee”**



doc_id	key_id
1	0
12	35
42	99

id	key
0	Java
35	Coffe
37	Mug

= coffe? NO

= coffe? YES

Example (2)

- The database server selects the proper data and provides results in form of a “rowset”.
  - In each row there is a valid entry extracted from the stored data
- The “/search” application iterates through the received data by means of a “cursor”
- The data is formatted/elaborated and the content of the “/search” result page is built
- The search result is sent back to the user

Example (3) (PHP)

```
<?php
```

The application composes the query

```
$query = “SELECT doc_id FROM key_doc_index,
  keywords WHERE key_doc_index.key_id =
  keywords.id AND keywords.key =
  $_REQUEST[“query”];”;
```

The query is sent to the db-server and a rowset containing the results is returned

```
$rowset = mysql_query($query);
```

```
while($row = mysql_fetch_row($rowset))
{
   //elaborate data
}
```

```
?>
```

The application elaborates the data

Technologies

- SQL is the standard query language
  - Legacy extensions are available
  - Extensions can empower the standard SQL but tie the application to a specific database server
- Interaction with the application server
  - ODBC, JDBC, ADODB, ...
  - Socket & Pipe
- Data access
  - Files stored on the server file-system
  - Files stored in ad-hoc partitions / disks

Performance and issues

- The database server is a critical component
- It is very difficult to optimize, if recognized as the bottleneck of a given architecture
- Performance depends on many factors:
  - The kind of database server
  - The Operating System of the host machine
  - The CPU power
  - The Memory availability and architecture
  - The I/O subsystem performance

Database servers

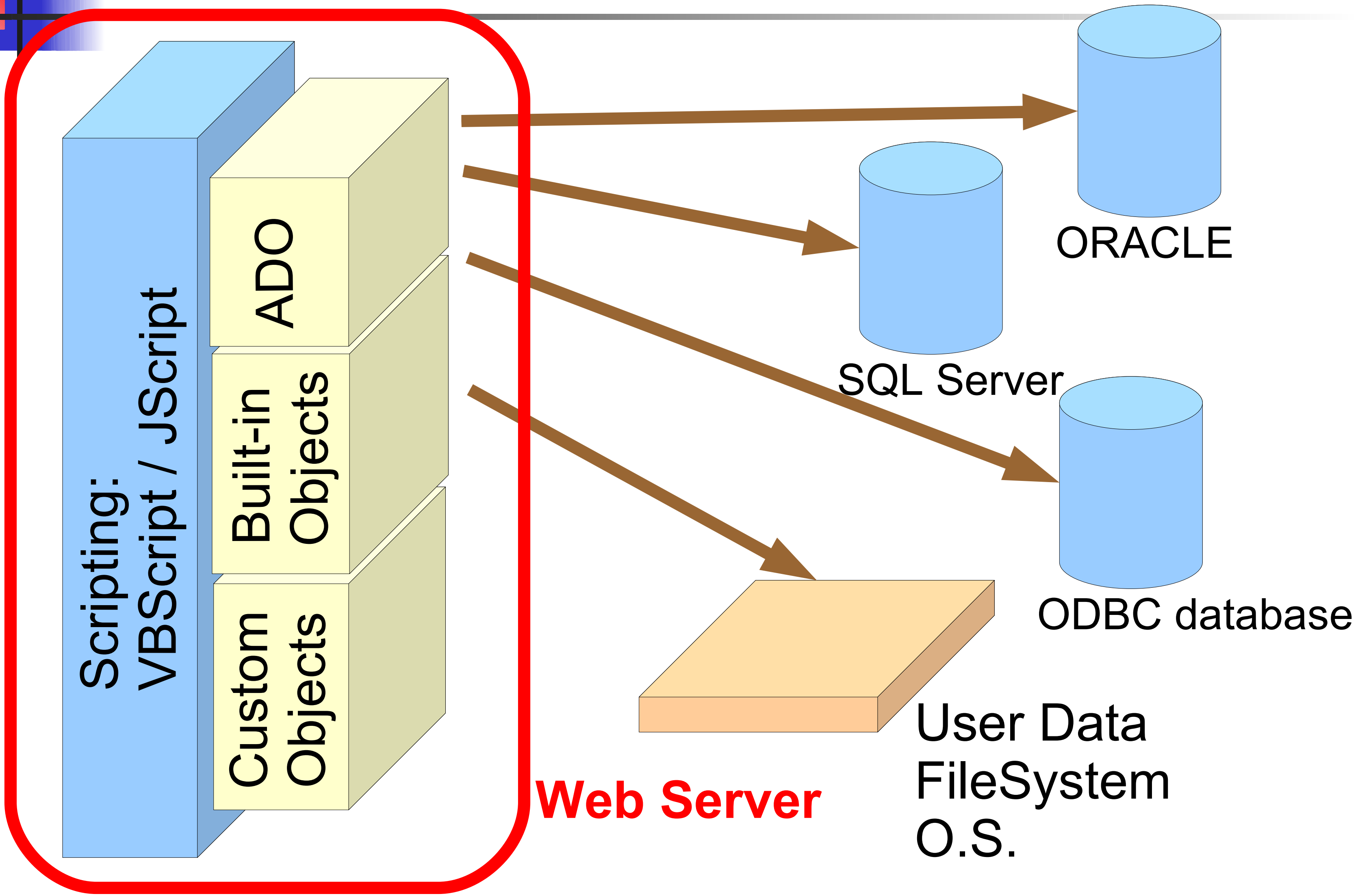
- Commercial solutions
  - Oracle
  - IBM DB2
  - Microsoft SQL server
- Open source
  - PostgreSQL
  - MySQL
  - Derby

Comprehensive platforms

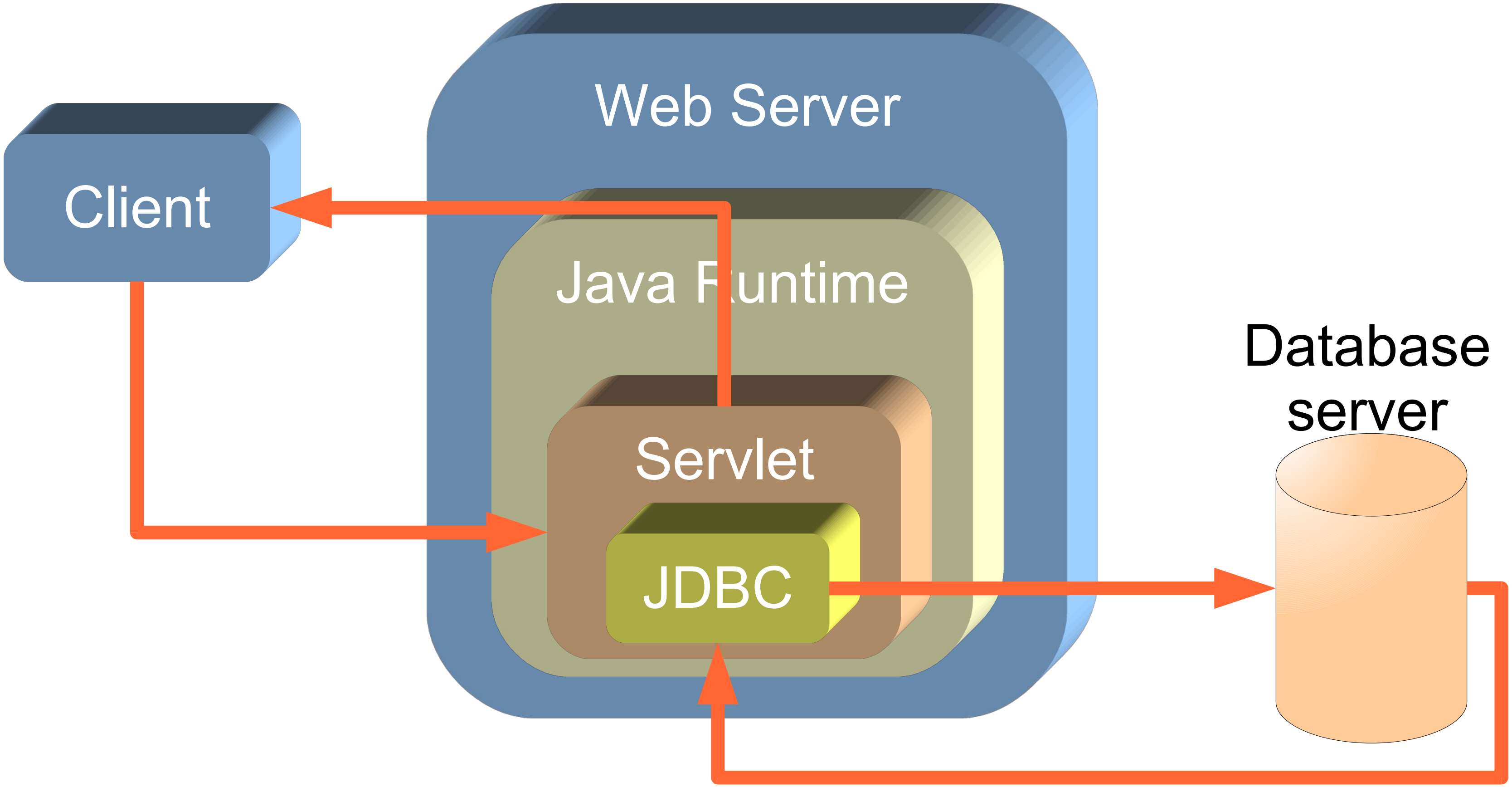
There are several solutions which allow to seamlessly use an Application Server together with a Database Server.

- Low-end applications
  - Microsoft ASP
  - PHP
  - Java Servlet or Java Server Pages (JSP)
- High-end applications
  - Java 2 Enterprise Edition
  - Microsoft .NET

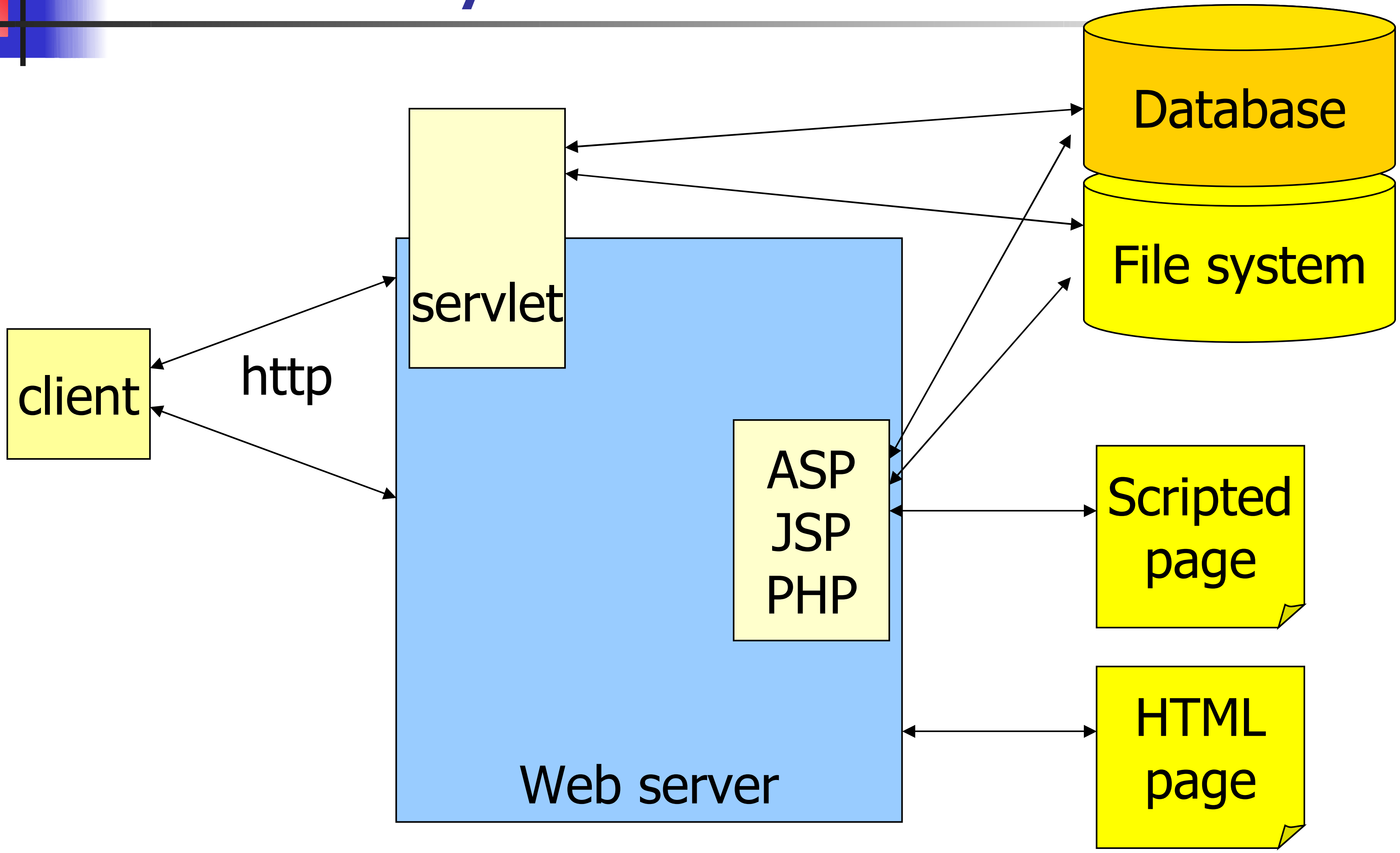
ASP


Scripting:
VBScript / JScript
ADO
Built-in Objects
Custom Objects
Web Server
ORACLE
SQL Server
ODBC database
User Data
FileSystem
O.S.

Servlet /JSP


Web Server
Client
Java Runtime
Servlet
JDBC
Database server

Summary



Database
File system
servlet
client
http
ASP
JSP
PHP
Scripted page
HTML page
Web server

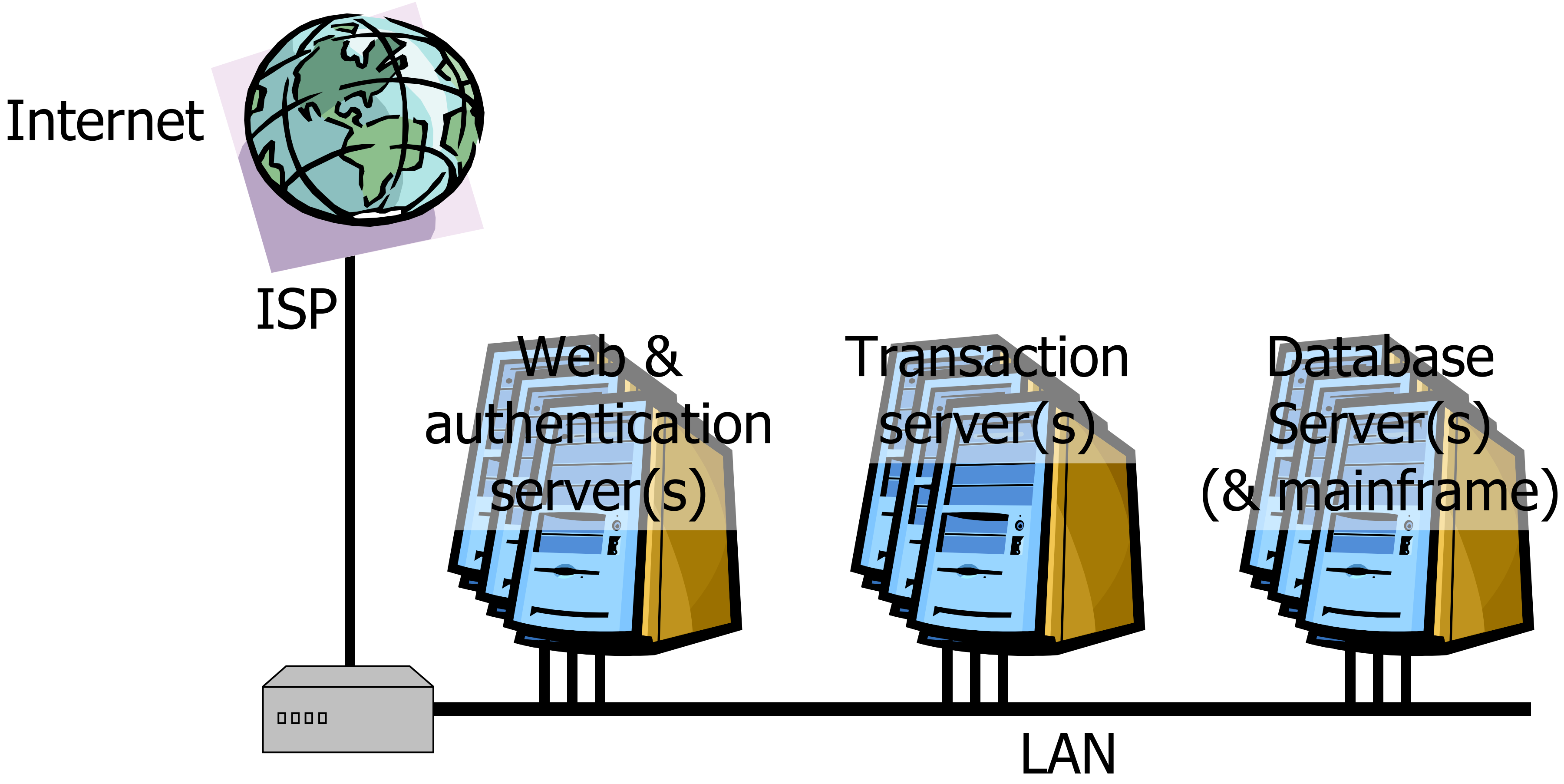
Outline

- Reference Architecture
- Web server
- Application server
- Database server
- Networks, proxy, cache
- Load balancer
- Enterprise frameworks

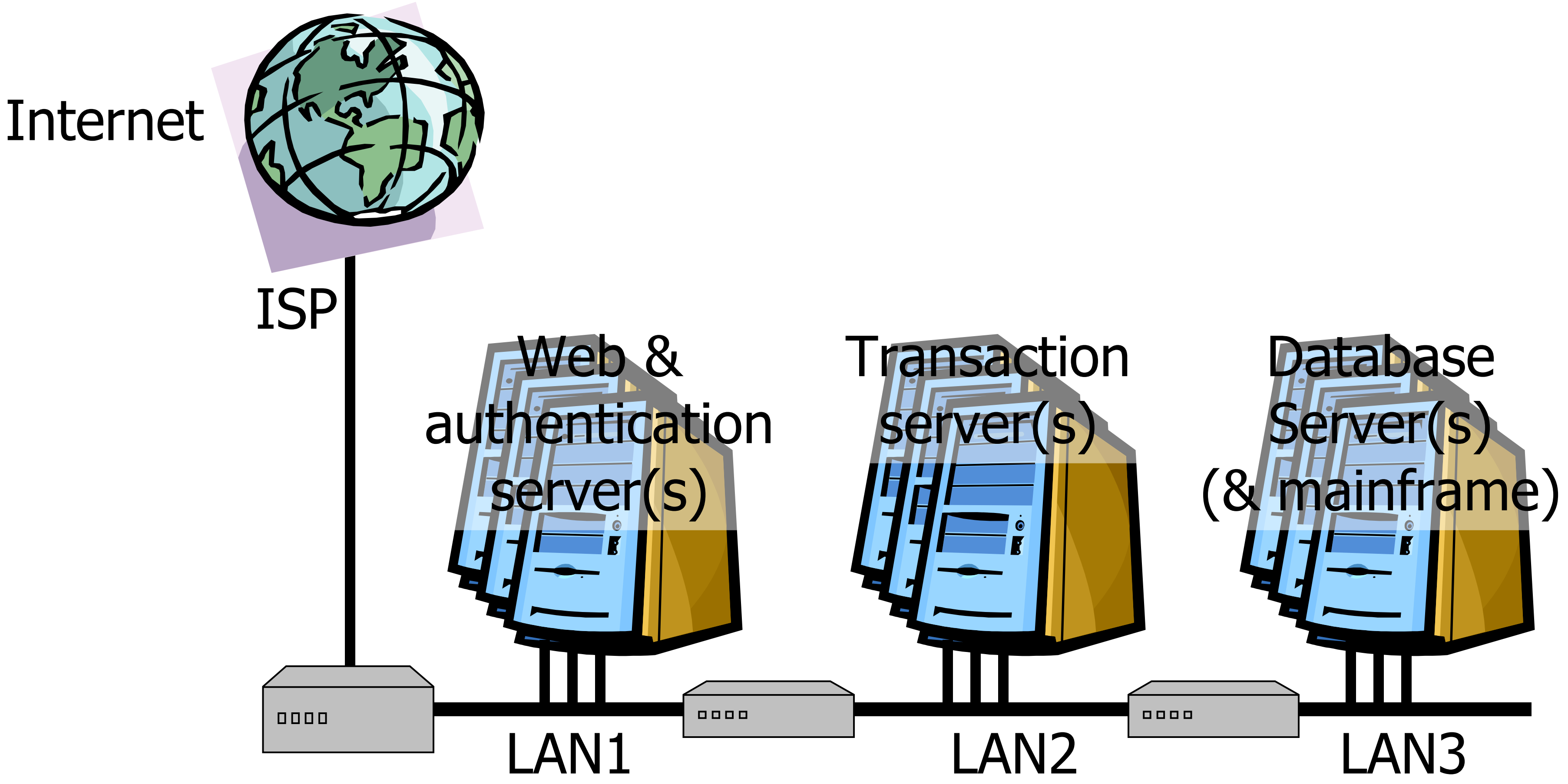
Network Infrastructure

- Allows to
  - Connect the site to the Internet
  - Inter-connect the servers on which the site relies
- Composed of:
  - Dedicated links, Local networks
  - Network devices
    - Switches, Routers and Firewalls
  - Performance-related elements
    - Proxies, Caches, ...

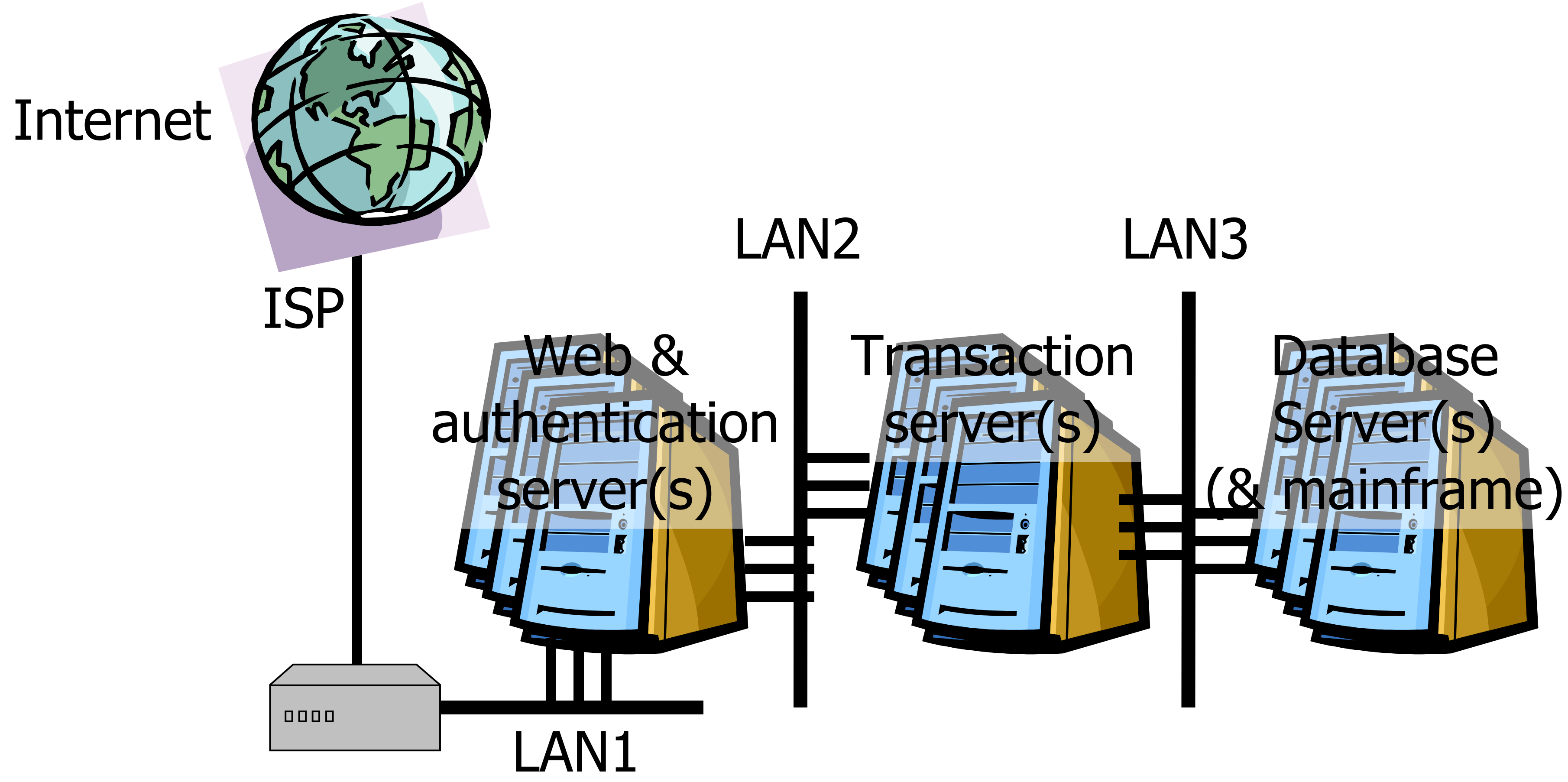
Example 1


Internet
ISP
Web & authentication server(s)
Transaction server(s)
Database Server(s) (& mainframe)
LAN

Example 2


Internet
ISP
Web & authentication server(s)
Transaction server(s)
Database Server(s) (& mainframe)
LAN1
LAN2
LAN3

Example 3


Internet
ISP
LAN2
LAN3
Web &
authentication
server(s)
Transaction
server(s)
Database
Server(s)
(& mainframe)
LAN1

Internet connection

- The link bandwidth limits the number of concurrent accesses
  - #Users/Time = Bandwidth / Page_size
- The connection reliability is critical
  - Reduced down-time
  - Minimum Bandwidth shall be granted

Local networks

- Usually based on
  - Ethernet (10 Mb/s)
  - Fast Ethernet (100 Mb/s)
  - Gigabit Ethernet (1000Mb/s)
- Can be segmented for facing high network traffic
  - Routers / Switches
  - Multiple network interfaces on servers

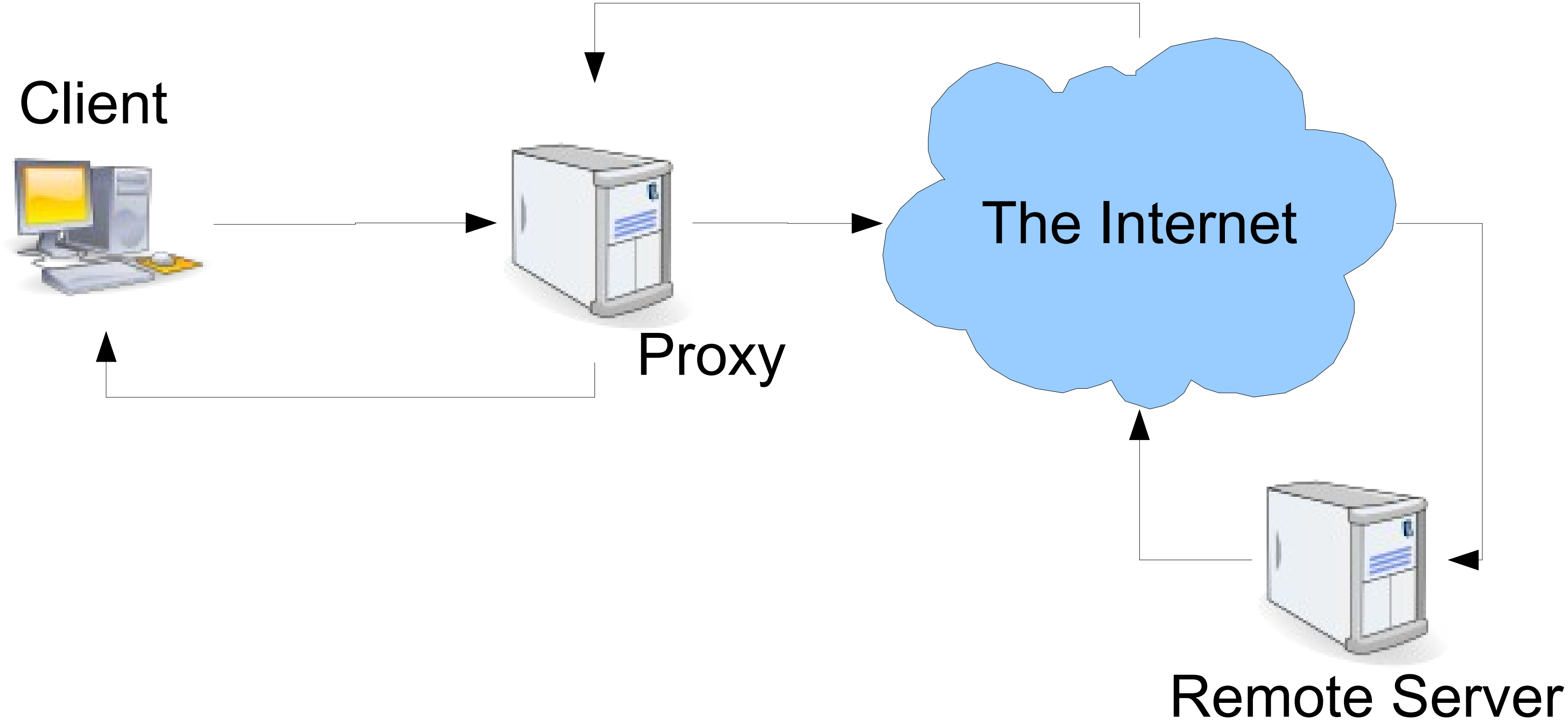
Firewall

- Filters out undesired network traffic
  - Outgoing
  - Incoming
- Protects from hacker's attacks and prevents data stealing
- Cannot address
  - Denial of service attacks
  - Applications vulnerabilities / bugs
- Increases the overall latency

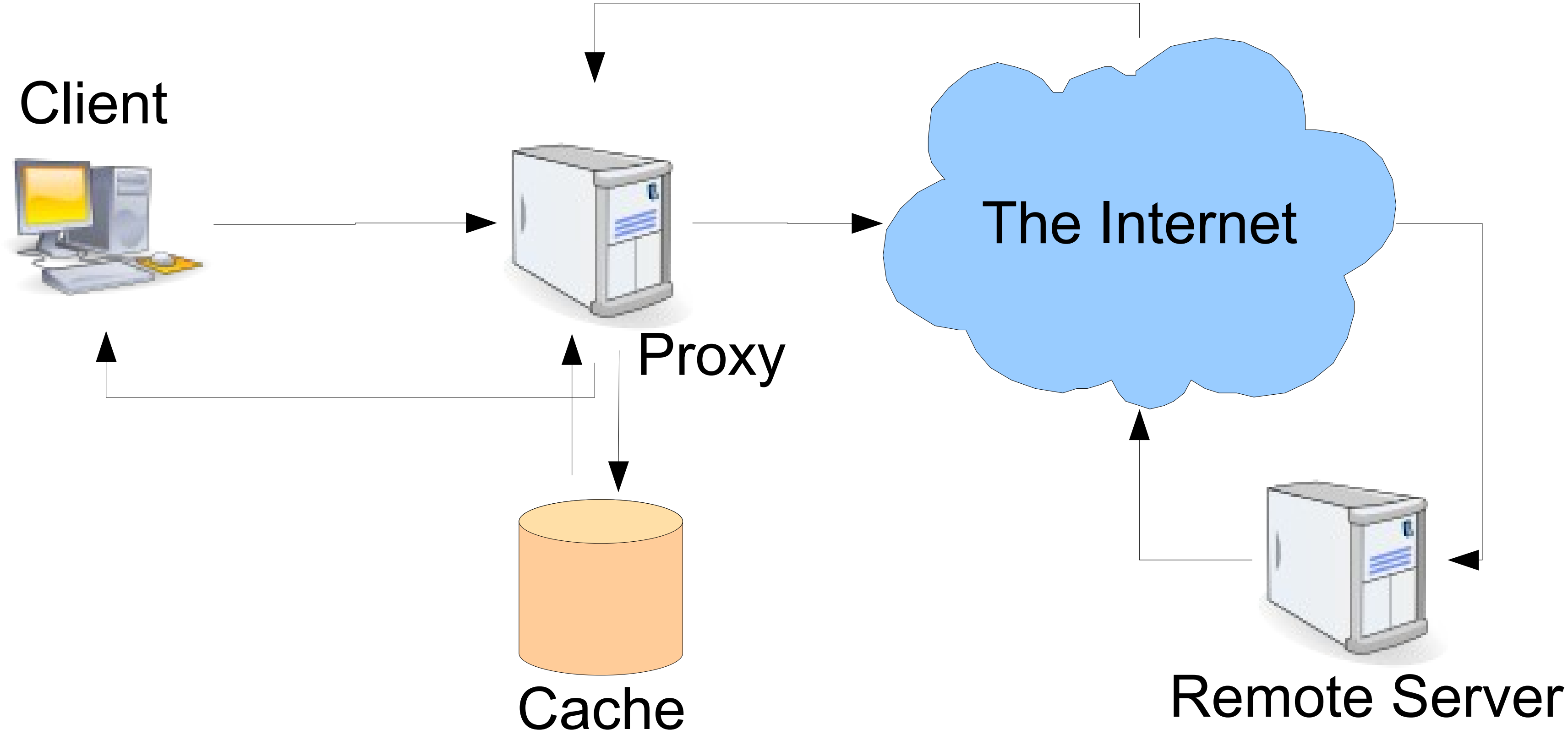
Proxy & Cache

- The two functionalities are often integrated into a single server/device
  - Proxy: intercepts the client http requests and mediates the access to the Internet thus hiding the client identity.
  - Cache: stores the results of client requests avoiding to access the remote server in case of repetitive requests.
- More useful for clients than for servers

Proxy & Cache (1)



Proxy & Cache (2)



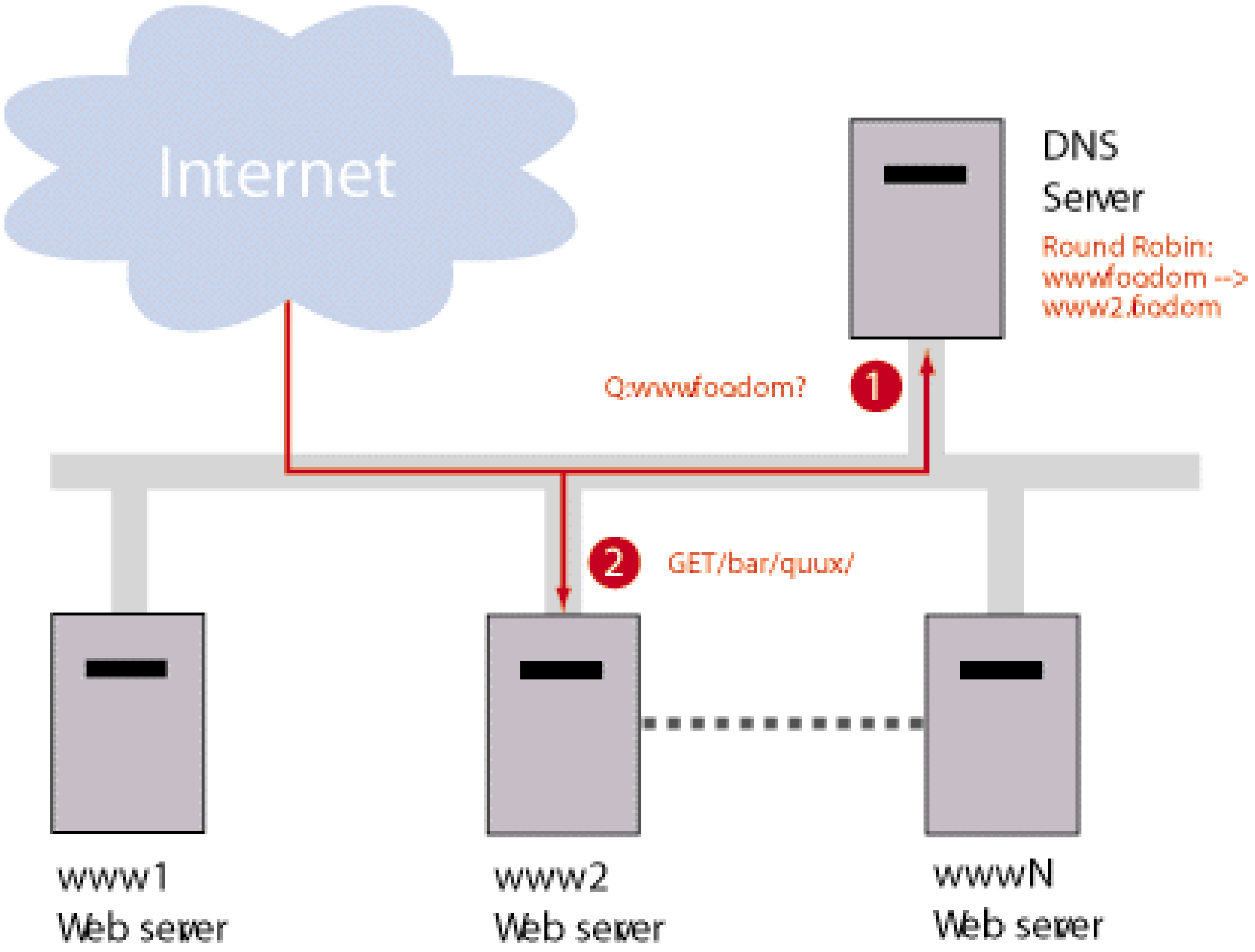
Outline

- Reference Architecture
- Web server
- Application server
- Database server
- Networks, proxy, cache
- Load balancer
- Enterprise frameworks

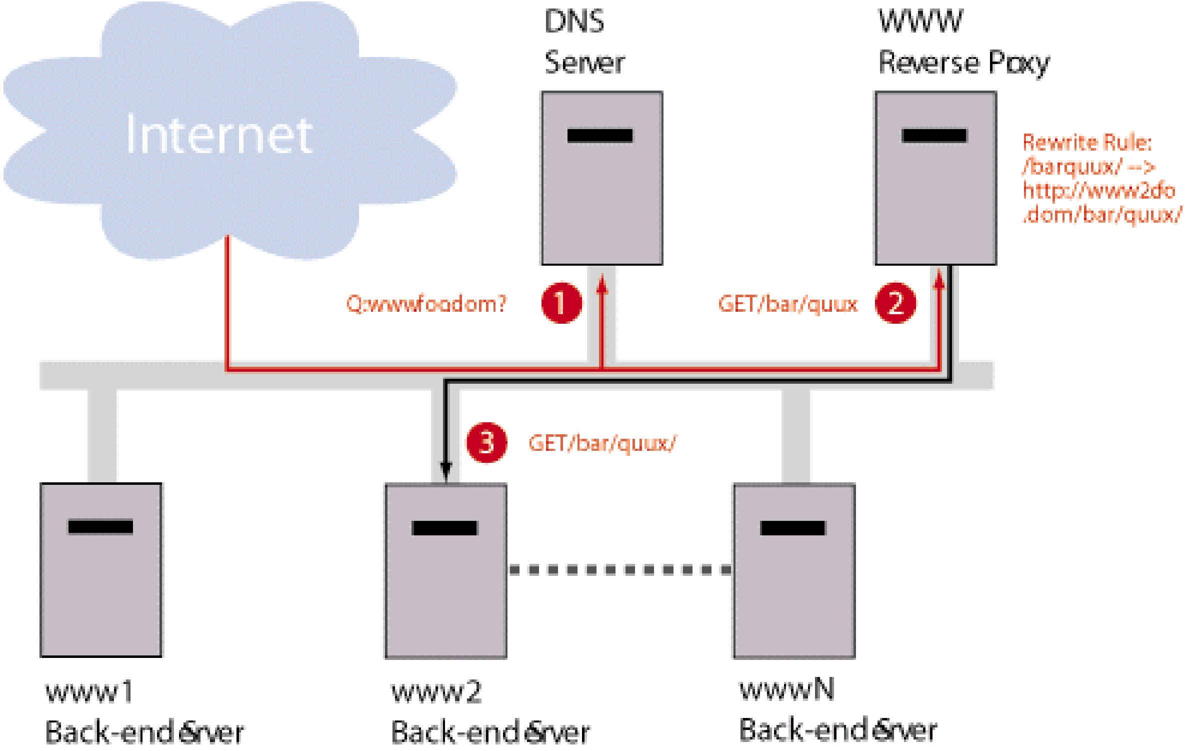
Load balancer

- Routes the client request towards a proper server (when more than one server is available)
- Continuously monitors each server workload
- Manages request queues
- Selects the most suited server

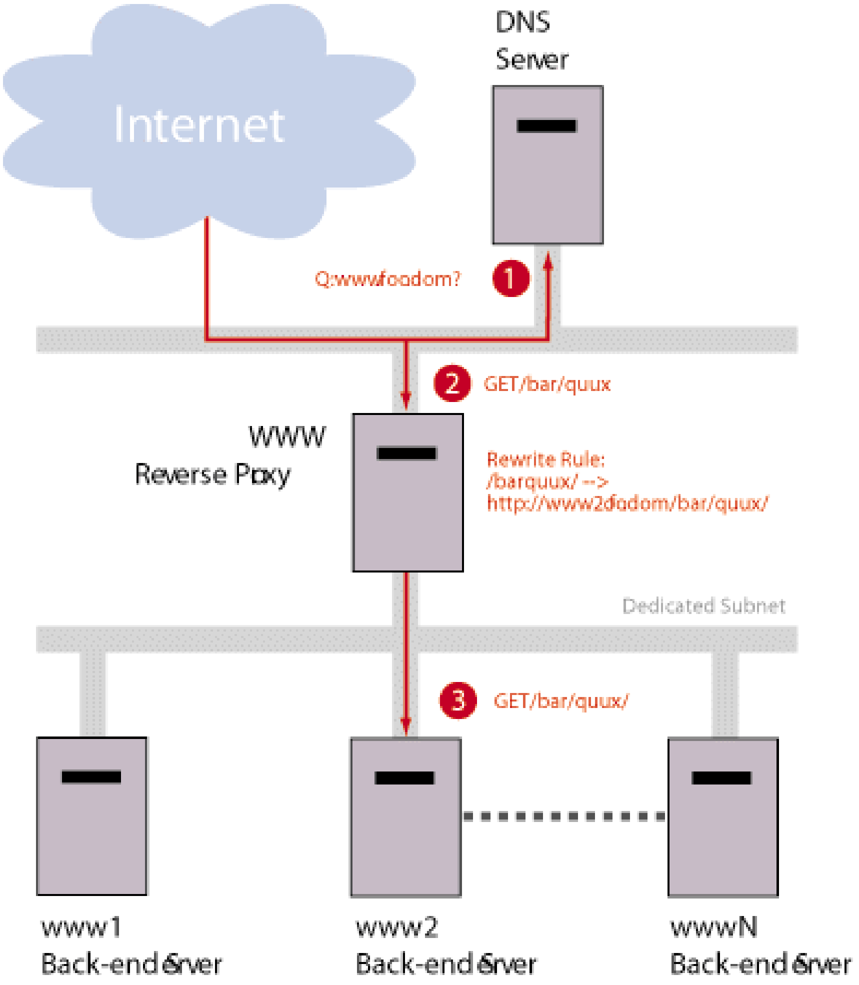
Example (round-robin DNS)



Example (reverse proxy, I)



Example (reverse proxy, II)



Outline

- Reference Architecture
- Web server
- Application server
- Database server
- Networks, proxy, cache
- Load balancer
- Enterprise frameworks

Requirements

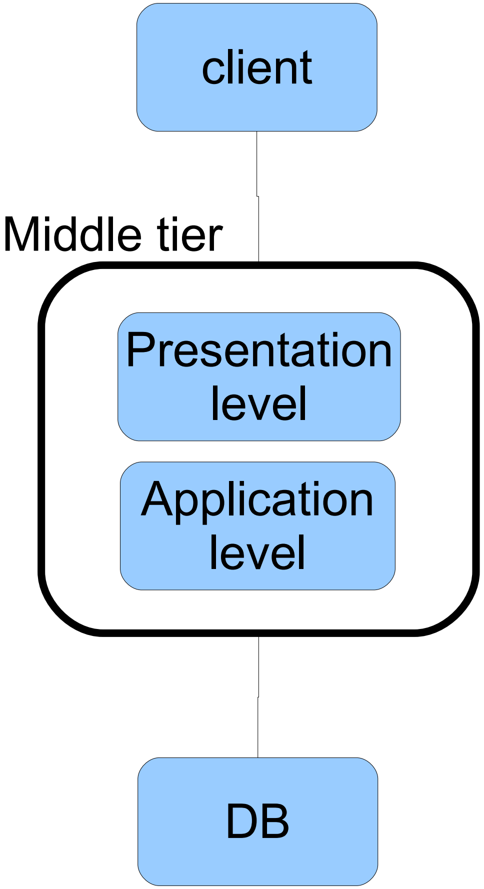
- Extension/improvement of already existing information systems with new functionalities
- High availability of the published content
- Security
- Reliability
- Scalability

Current market

- Over 1 billion dollars in 2001
- Both small and big vendors
- More than 25 different application servers available
  - IBM
  - Microsoft
  - Sun
  - Oracle
  - Netscape
  - BEA
  - …

Main features

- From the developer perspective the application server forces the separation between
  - business logic
  - presentation logic
  - database logic
- Real 3-tiers architecture with clear separation between the database and the application logic.



Addressed problems

- concurrency
- providing access to all possible production databases
- network connection management
- database connection pooling
- legacy database support
- designing a management console
- clustering support
- load balancing
- fail-over
- extensibility of development framework and performance

Pros

- Can simplify the development of new systems
  - Many technical issues already addressed by the framework
  - Developers can focus their attention on the application-specific features/issues
  - Allow to easily build and manage very complex systems

Cons

- Considerable start-up effort
  - Can be quite complex to use
  - Requires specialized developers (higher formative needs)
  - Training (2-6 month)
  - Shall adhere to the programming standards defined by the framework

Main features

- Support for business and application logic
  - Technologies: COM, CORBA, Java-specific RMI
  - Framework: Microsoft .NET, Enterprise JavaBeans.
  - Multi-threaded access
  - Database connection pooling
  - Clustering support, load balancing and fail-over features
  - Transaction Integrity.
  - Connectivity to legacy systems like mainframes and older transaction and database systems.
  - Secure transactions and support for SSL/HTTPS

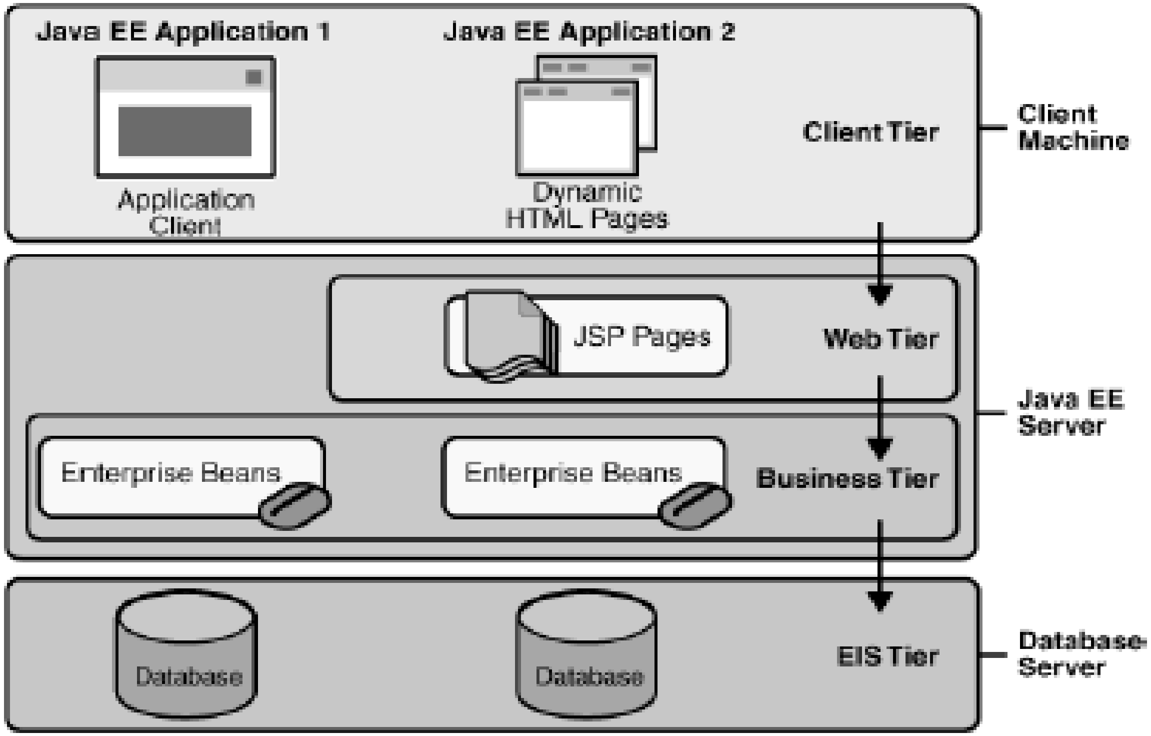
Other features

- Web deployment: Internet, Intranet, ...
- Transaction processing monitors
- Transaction processing and performance oriented features
- Support for working with other application servers
- Integration with development tools and features oriented towards accelerating development
- Integration with Enterprise resource planning packages like SAP/R3 and Peoplesoft.
- ...

J2EE

- Multi-tiered application model
- Based on the Java technology
- The various components usually run on different machines
  - Client-tier components run on the client machine.
  - Web-tier components run on the Java EE server.
  - Business-tier components run on the Java EE server.
  - Enterprise information system (EIS)-tier software runs on the EIS server.

J2EE (2)


Java EE Application 1
Java EE Application 2
Application Client
Dynamic HTML Pages
Client Tier
Client Machine
JSP Pages
Web Tier
Enterprise Beans
Enterprise Beans
Business Tier
Java EE Server
Database
Database
EIS Tier
Database Server

J2EE Client Tier

- Web Clients (thin client)
  - Dynamic pages + Web Browser
- Applets
  - Small Java application embedded in a web page
  - Execute on the client virtual machine
  - They often require a Java Plugin and they pose some serious security issues (applet sandbox)

J2EE Client Tier (2)

- Application Client (thick client)
  - Java application running on the client machine
  - Richer user interface
  - Can directly interface Enterprise beans (otherwise can interface Servlets over http)

J2EE Web Tier

- Servlets
  - Java programming language classes that dynamically process requests and construct responses
- Java Server Pages (JSP)
  - text-based documents that execute as servlets but allow a more natural approach to creating static content
- Java Server Faces (JSF)
  - provides a user interface component framework for web applications (based on servlets and JSPs)

J2EE Business Tier

- logic that solves or meets the needs of a particular business domain such as banking, retail, etc.
- Beans
  - body of code having fields and methods to implement modules of business logic
  - 3 kind of Beans
    - Session Beans
    - Message Beans
    - Entity Beans (Java Persistence Entities)

J2EE Beans

- Session beans
  - Represent a transient conversation with a client
  - When the client finishes executing, the session bean and its data are gone
- Message beans
  - Allow a business component to receive messages asynchronously
  - Session bean + message listener

J2EE Beans (2)

- Entity beans
  - represent persistent data stored in one row of a database table
  - If the client terminates, or if the server shuts down, the persistence manager ensures that the entity data is saved

J2EE EIS Tier

Enterprise Information System tier

- handles EIS software including:
  - Enterprise resource planning (ERP)
  - Mainframe transaction processing
  - Database systems
  - Legacy information systems

Available platforms

- Apache: Tomcat
- Apple: WebObjects
- ATG: Dynamo
- BEA: WebLogic
- Borland: Enterprise Server
- Caucho: Resin
- Desiderata Software: Blazix
- Fujitsu Siemens Computers: BeanTransactions
- Fujitsu Software Corp.: Interstage
- Gefion Software: LiteWebServer
- HHPN: XliRAD
- Hitachi: Cosminexus
- IBM: Websphere

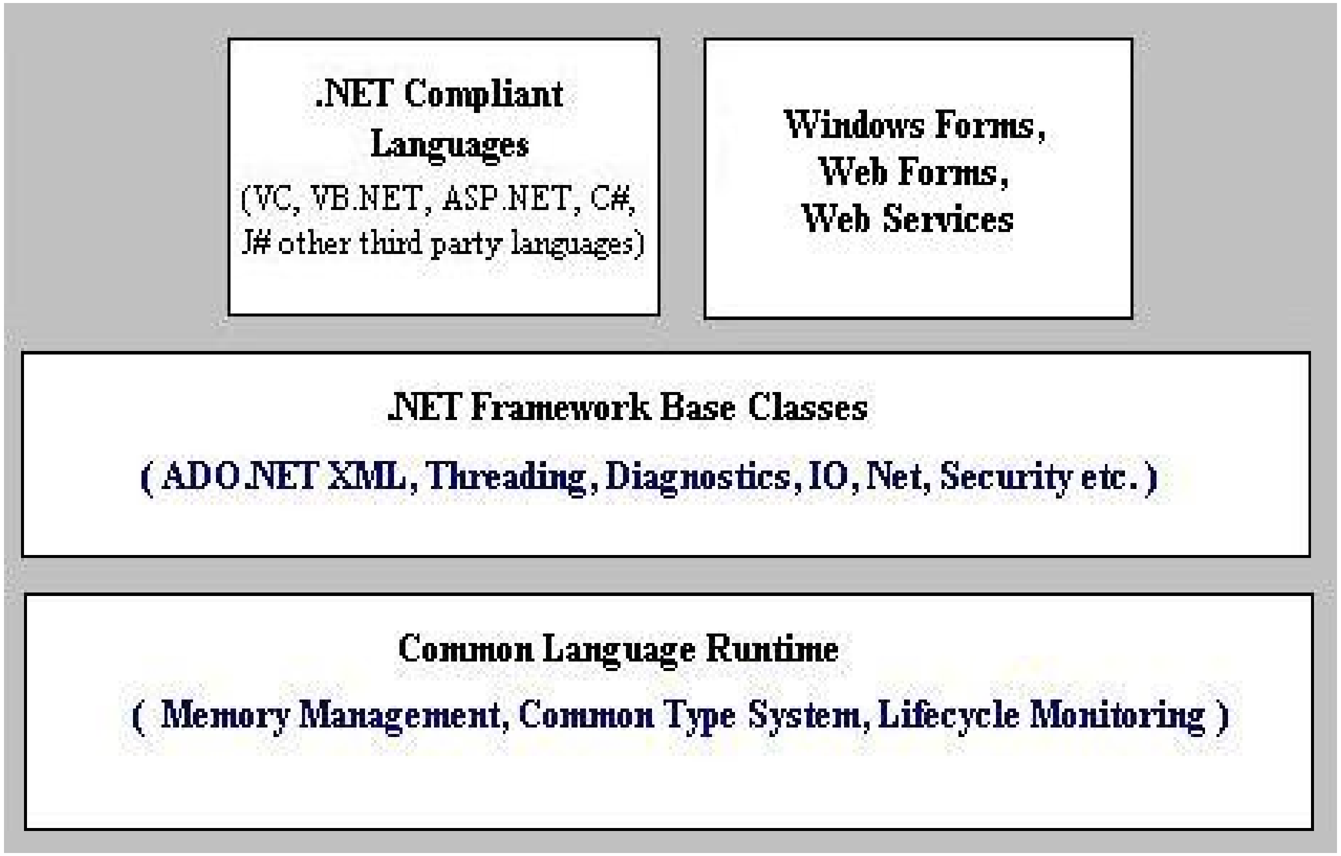
Available platforms (2)

- Interactive Business Solutions: EAS
- IONA: Orbix ASP 6.0
- Ironflare: Orion
- Jboss: Jboss
- Jetty: Mort Bay Consulting
- Macromedia: JRun Server
- New Atlanta Communications: ServletExec
- Novell: exteNd
- ObjectWeb: JonAS
- OpenConnect
- OC://WebConnect
- Oracle Application Server 10g
- Persistence: Power Tier for J2EE

Available platforms (3)

- Pramati Technologies: Pramati Server
- SAP AG: Web Application Server
- Secant: ModelMethods
- Sun Microsystems: Sun Java System Application Server
- Sybase: EAServer
- TmaxSoft: JEUS
- Together: Enhydra Server
- TradeCity Cybersoft: Rexip Appserver
- Trifork: Enterprise Application Server

Microsoft .NET framework



.NET Compliant Languages
(VC, VB.NET, ASP.NET, C#, J# other third party languages)

Windows Forms,
Web Forms,
Web Services

.NET Framework Base Classes
(ADO.NET XML, Threading, Diagnostics, IO, Net, Security etc.)

Common Language Runtime
(Memory Management, Common Type System, Lifecycle Monitoring)

Microsoft .NET framework (2)

- Can be exploited by .NET compliant languages (language neutral)
  - C#, C++
  - JScript
  - Visual Basic
- ASP .NET
  - Class libraries for web-based services
- Introduces Web Forms, Web Services and Windows Forms
  - Web forms -> web interfaces
  - Windows forms -> windows GUIs

Microsoft .NET framework (3)

- .NET Framework Base Classes
  - Common class libraries
  - Can be used by any .NET language
  - Provide many ready-to-use functionalities
- CLR: common language runtime
  - Similar to the Java Virtual Machine
  - Converts the MSIL code (Microsoft Intermediate Language) into executable code

Web Forms

- Provide a unified tool for designing web interfaces
  - Can use HTML, DHTML or WML to draw the controls on the web page depending on the browser capability
  - Incorporate the logic behind each control
  - It is possible to associate a specific code to each GUI element (the code is executed on the server)

Web Forms (2)

- A Web form is composed of 2 parts:
  - A template: HTML based layout information for all the GUI elements
  - A component: contains the logic to be hooked to each GUI element
- The GUI is rendered on the client side
- The hooked code is executed on the server side

Web Forms (3)

- Web forms render differently depending on the browser capabilities:
  - Internet Explorer (DHTML)
  - Netscape / Mozilla (HTML)
  - WAP device (WML)

Available platforms

- Microsoft IIS + .NET framework