ATI – Arsitektur Aplikasi Web

anton@ukdw.ac.id

Internet & Web Basics

Internet

- What is Internet?
  - **A world-wide network of computer networks**
- Internet is huge client server system
- A Server runs continuously, **waiting** to be contacted by a Client
  - Each Server provides certain **services**
  - Services include providing *web pages => web application => web services*
- A Client will **send** a *message* to a Server **requesting** the service provided by that server
  - The client will usually provide *some information, parameters*, with the request

Web

- **Tim Berners-Lee** at CERN proposed the Web in 1989
  - **Purpose**: to allow scientists to have access to many databases of scientific work through their own computers
- From document (**linier media**) to hypertext (**hyper media**)
  - Ex of Linier media is a book, pages, document, cassete
    - We'll call them **documents**
  - Hypermedia – media that provide point to the need
    - Examples: CD, DVD, Webpages
    - Web provides **Hyperlink**

Protokol Web: **HTTP**

- **Hypertext** Transport Protocol (RFC 1945)
  - Oleh Tim Berners-Lee, 1990
- Protocol that used to communicate between web browsers and web servers
- Using TCP port **80** (default)
- This protocol supports ***hypermedia*** files
- HTTP 1.0 becomes 1.1 by IETF (RFC 2616)

HTTP Session

- When client and server communicate, their connection has to be maintenance…
- HTTP provides session
- A basic **HTTP session** has four phases:
  - Client opens the connection
  - Client makes the request
  - Server sends a response
  - Server closes the connection

HTTP Stateless

- But HTTP is "**stateless**"
  - HTTP is a **stateless protocol**, which means that once a server has delivered the requested data to a client, the connection is broken, and the server retains no memory of what has just taken place

- No information that retain
- Solution? **Cookies & Session**

HTTP (2)

- HTTP is **request – response**:
  - HTTP client (**user agent**) request s to HTTP server and Server will response
- The basic different of HTTP/1.1 and HTTP/1.0 is their **persistent connection**
- HTTP/1.0 is **not** using persistent connection
  - Header = Connection: close
- HTTP/1.1 is using **persistent** connection
  - Header = Connection: Keep-Alive
  - But client can set header manually (header = Connection: close).


```
CONNECT www.bad.carbonwind.net:443 HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.2; SV1; .NET CLR 1.1
.4322)
Host: www.bad.carbonwind.net
Content-Length: 0
Proxy-Connection: Keep-Alive
Pragma: no-cache

HTTP/1.1 200 Connection established
Via: 1.1 LAB4ISA
Connection: Keep-Alive
Proxy-Connection: Keep-Alive
```

HTTP Message

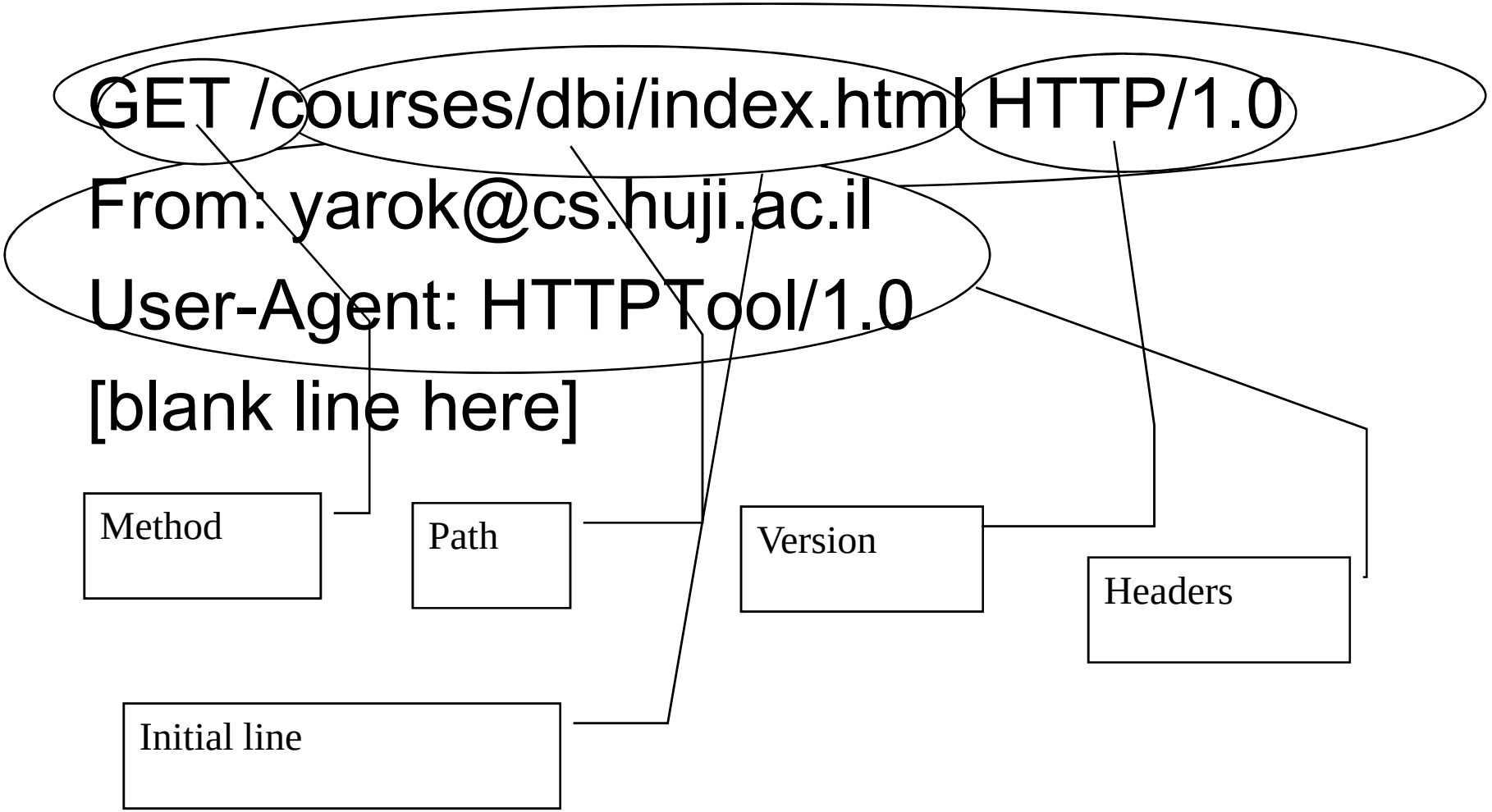
```
HTTP-message    = Request | Response ; HTTP/1.1 messages

generic-message = start-line
                  *(message-header CRLF)
                  CRLF [ message-body ]

start-line      = Request-Line | Status-Line

message-header  = field-name ":" [ field-value ]

Request-Line    = Method SP Request-URI SP HTTP-Version CRLF

Method          = "OPTIONS"                ; Section 9.2
                | "GET"                    ; Section 9.3
                | "HEAD"                   ; Section 9.4
                | "POST"                   ; Section 9.5
                | "PUT"                    ; Section 9.6
                | "DELETE"                 ; Section 9.7
                | "TRACE"                  ; Section 9.8
                | "CONNECT"                ; Section 9.9
                | extension-method
```

'host:' mandatory

Request

- A request consists of:
  - Initial line
  - Headers
  - Blank line
  - Message body

Request Example



HTTP Request Methods

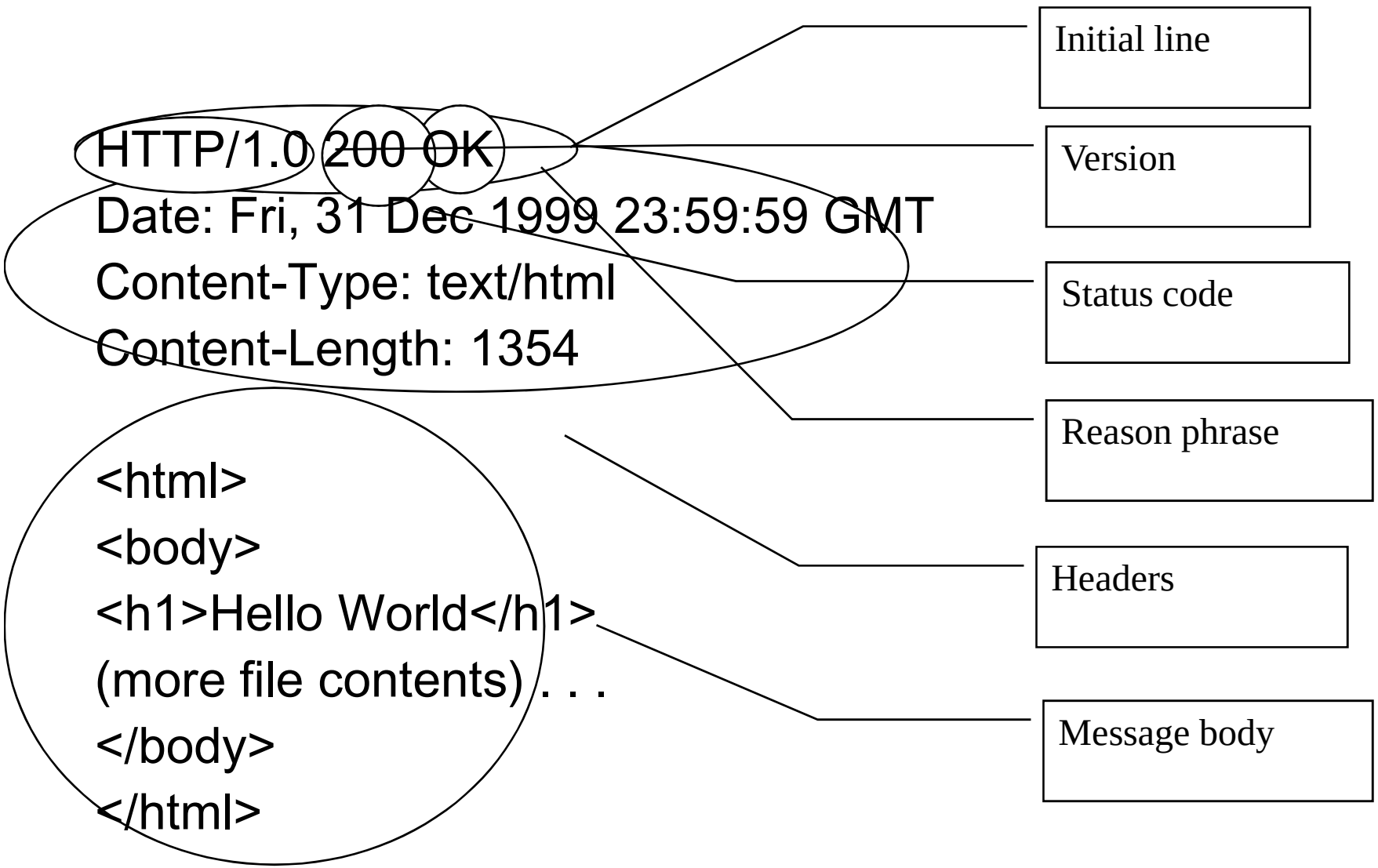
- **GET** - Fetch a document
  - Ex: URL variabel
- **POST** - Execute the document, using the data in body document
  - Ex: form submit
- **HEAD** - Fetch just the header of the document
  - Ex: date modified
- **PUT** - Store a new document on the server
  - Ex: upload using WebDAV
- **DELETE** - Remove a document from the server

HTTP Response status

Response Code	HTTP Operation	Response Body Contents	Description
200	GET, PUT, DELETE	Resource	No error, operation successful.
201 Created	POST	Resource that was created	Successful creation of a resource.
202 Accepted	POST, PUT, DELETE	N/A	The request was received.
204 No Content	GET, PUT, DELETE	N/A	The request was processed successfully, but no response body is needed.
301 Moved Permanently	GET	XHTML with link	Resource has moved.
303 See Other	GET	XHTML with link	Redirection.
304 Not Modified	conditional GET	N/A	Resource has not been modified.
400 Bad Request	GET, POST, PUT, DELETE	Error Message	Malformed syntax or a bad query.
401 Unauthorized	GET, POST, PUT, DELETE	Error Message	Action requires user authentication.
403 Forbidden	GET, POST, PUT, DELETE	Error Message	Authentication failure or invalid Application ID.
404 Not Found	GET, POST, PUT, DELETE	Error Message	Resource not found.
405 Not Allowed	GET, POST, PUT, DELETE	Error Message	Method not allowed on resource.

406 Not Acceptable	`GET`	Error Message	Requested representation not available for the resource.
408 Request Timeout	`GET, POST`	Error Message	Request has timed out.
409 Resource Conflict	`PUT, PUT, DELETE`	Error Message	State of the resource doesn't permit request.
410 Gone	`GET, PUT`	Error Message	The URI used to refer to a resource.
411 Length Required	`POST, PUT`	Error Message	The server needs to know the size of the entity body and it should be specified in the Content Length header.
412 Precondition failed	`GET`	Error Message	Operation not completed because preconditions were not met.
413 Request Entity Too Large	`POST, PUT`	Error Message	The representation was too large for the server to handle.
414 Request URI too long	`POST, PUT`	Error Message	The URI has more than 2k characters.
415 Unsupported Type	`POST, PUT`	Error Message	Representation not supported for the resource.
416 Requested Range Not Satisfiable	`GET`	Error Message	Requested range not satisfiable.
500 Server Error	`GET, POST, PUT`	Error Message	Internal server error.
501 Not Implemented	`POST, PUT, DELETE`	Error Message	Requested HTTP operation not supported.
502 Bad Gateway	`GET, POST, PUT, DELETE`	Error Message	Backend service failure (data store failure).
505	`GET`	Error Message	HTTP version not supported.

Response Example



HTTP Environment Variables

SERVER_PROTOCOL	HTTP version as defined on the request line following HTTP method and URL.
SERVER_PORT	Server port used for submitting the request, set by the server based on the connection parameters.
REQUEST_METHOD	HTTP method as defined on the request line.
PATH_INFO	Extra path information in the URL. For example, if the URL is http://mysite.org/cgi-bin/zip.cgi/test.html, and http://mysite.org/cgi-bin/zip.cgi is the location of a CGI script, then /test.html is the extra path information.
PATH_TRANSLATED	Physical location of the CGI script on the server. In our example, it would be /www/cgi-bin/zip.cgi assuming that the server is configured to map the /cgi-bin path to the /www/cgi-bin directory.
SCRIPT_NAME	Set to the path portion of the URL, excluding the extra path information. In the same example, it's /cgi-bin/zip.cgi
QUERY_STRING	Information that follows the '?' in the URL.

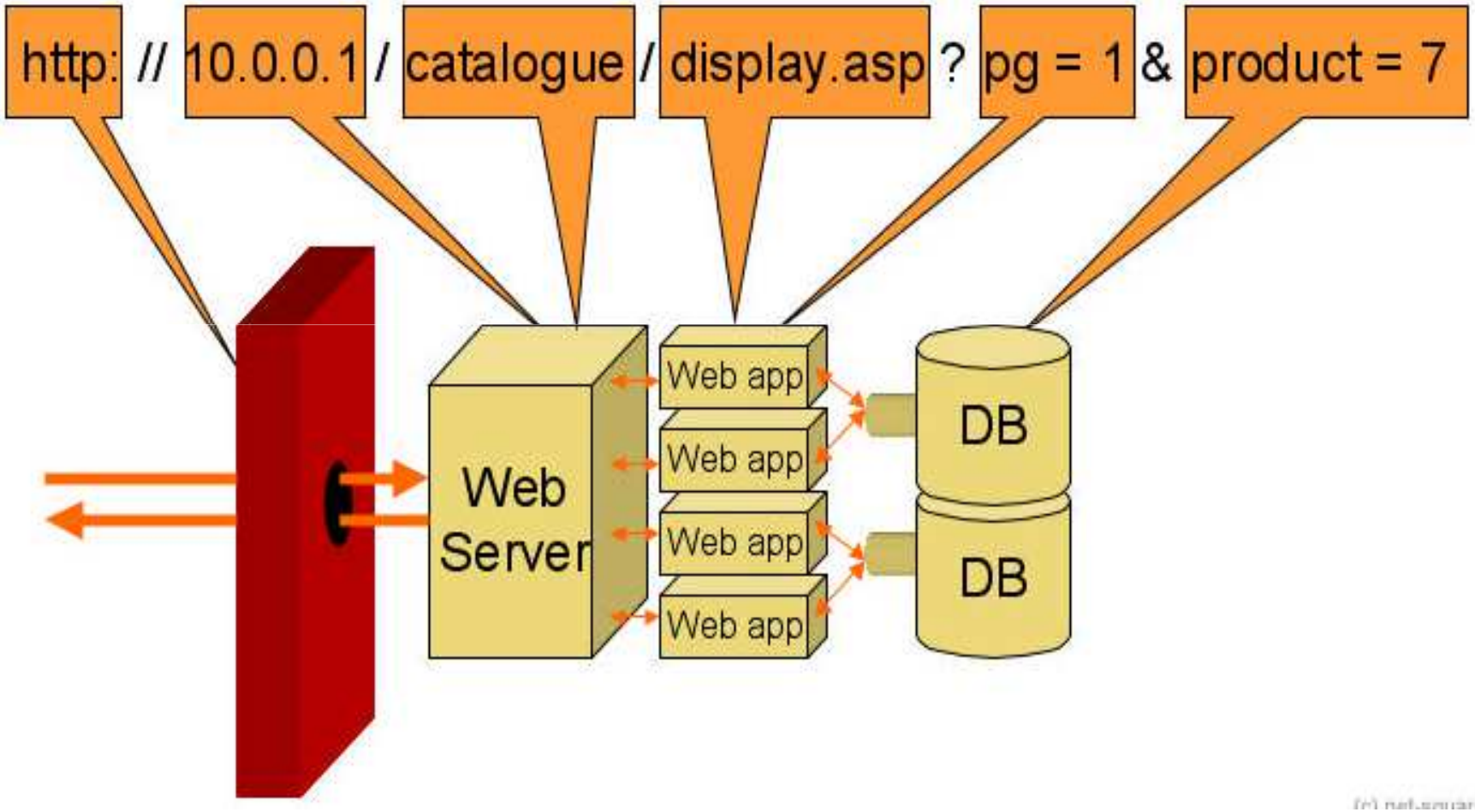
Web Server Basics

- The main job of a Web server computer is to **respond** to requests from Web client computers
- Three main elements of a Web server:
  - Hardware
  - Operating system software
  - Web server software

Web Server basic capabilities

- Virtual hosting: multiple domain
- Address mapping: map request address to the actual file on server
- Authentication: by password, host credentials
- Handling MIME (multi purpose internet mail extensions)
- Caching support
- Security support

Mapping URL Sistem Web



Web Server Software

- The most popular Web server programs are:
  - Apache HTTP Server
  - Microsoft Internet Information Server (IIS)
- Netcraft
  - Accumulates popularity rankings

Developer	December 2010	Percent	January 2011	Percent	Change
Apache	661,722	66.61%	662,184	66.62%	0.01
Microsoft	163,589	16.47%	161,219	16.22%	-0.25
nginx	58,705	5.91%	60,782	6.12%	0.21
Google	19,623	1.98%	20,524	2.06%	0.09

Web Server Hardware Architectures

- **Server farms**
  - Large collections of servers
- **Centralized architecture**
  - Uses a few very large and fast computers
- **Distributed/decentralized architecture**
  - Uses a large number of less powerful computers
  - Divides the workload among them

Web Browsers

- Browsers are **clients**
- **Mosaic** - NCSA (Univ. of Illinois), in early 1993
  - First to use a GUI, led to explosion of Web use
  - Initially for X-Windows, under UNIX, but was ported to other platforms by late 1993
- Most **requests** are for existing **documents**, using HTTP
  - But some requests are for *program execution*, with the output being returned as a **document**

Web Tier

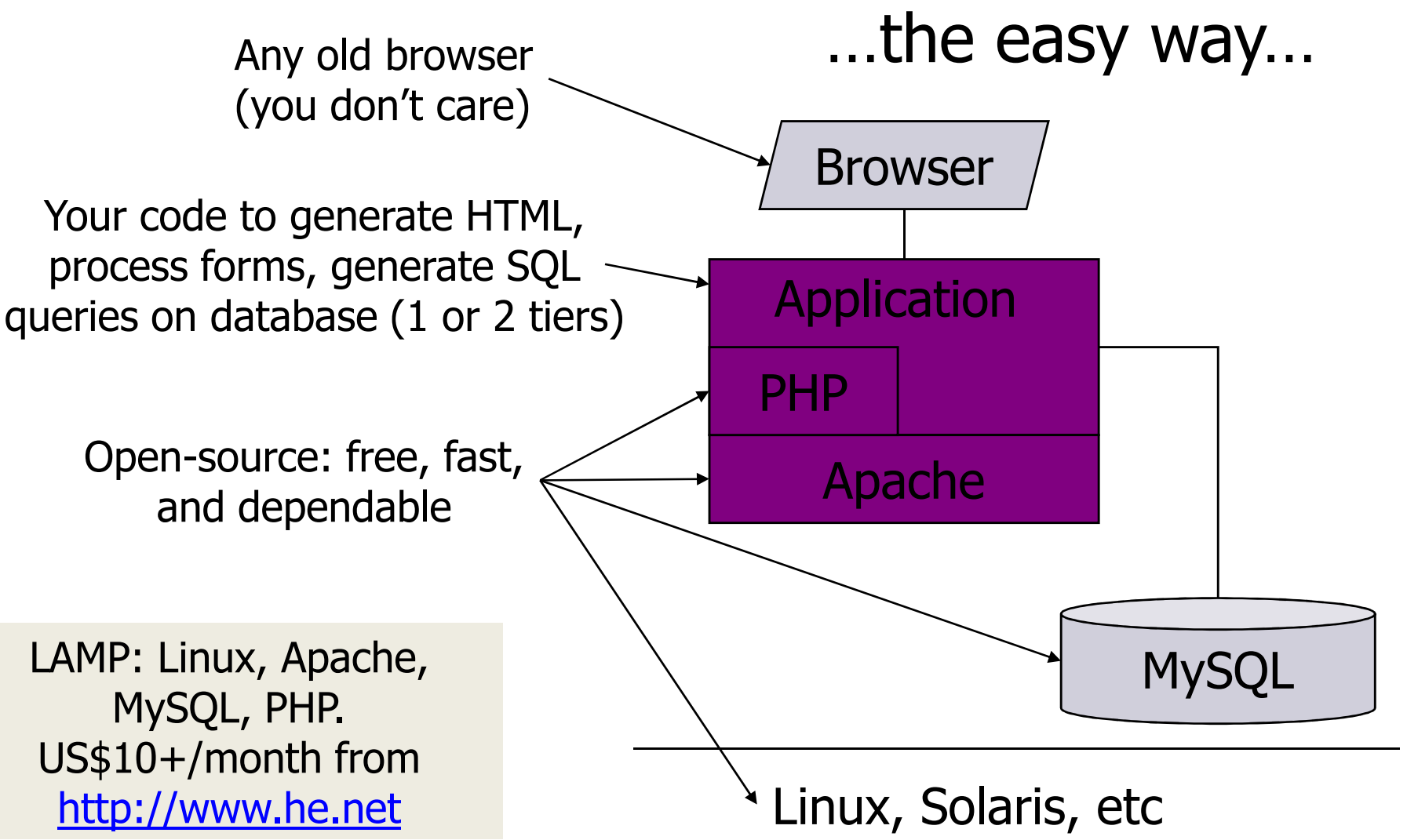
Web Architecture

- **Layering Aspect**
  - "Separation of concerns"
  - How many concurrent users are you serving?
  - Shared needs among multiple applications?
- **Data Aspect**
  - What kind(s) of data are you delivering?
    - Structured vs non structured
    - On-demand vs. real-time
  - What are the bandwidth requirements?
    - Size & nature of data
    - audience concerns

Tiers…?

- A “tier” can be hardware, software, or **logical** in layer.
- Therefore:
  - A non-application-specific client (such as a web **browser**) is not a tier
  - A **database** with no overlying data access layer is not really considered a tier either

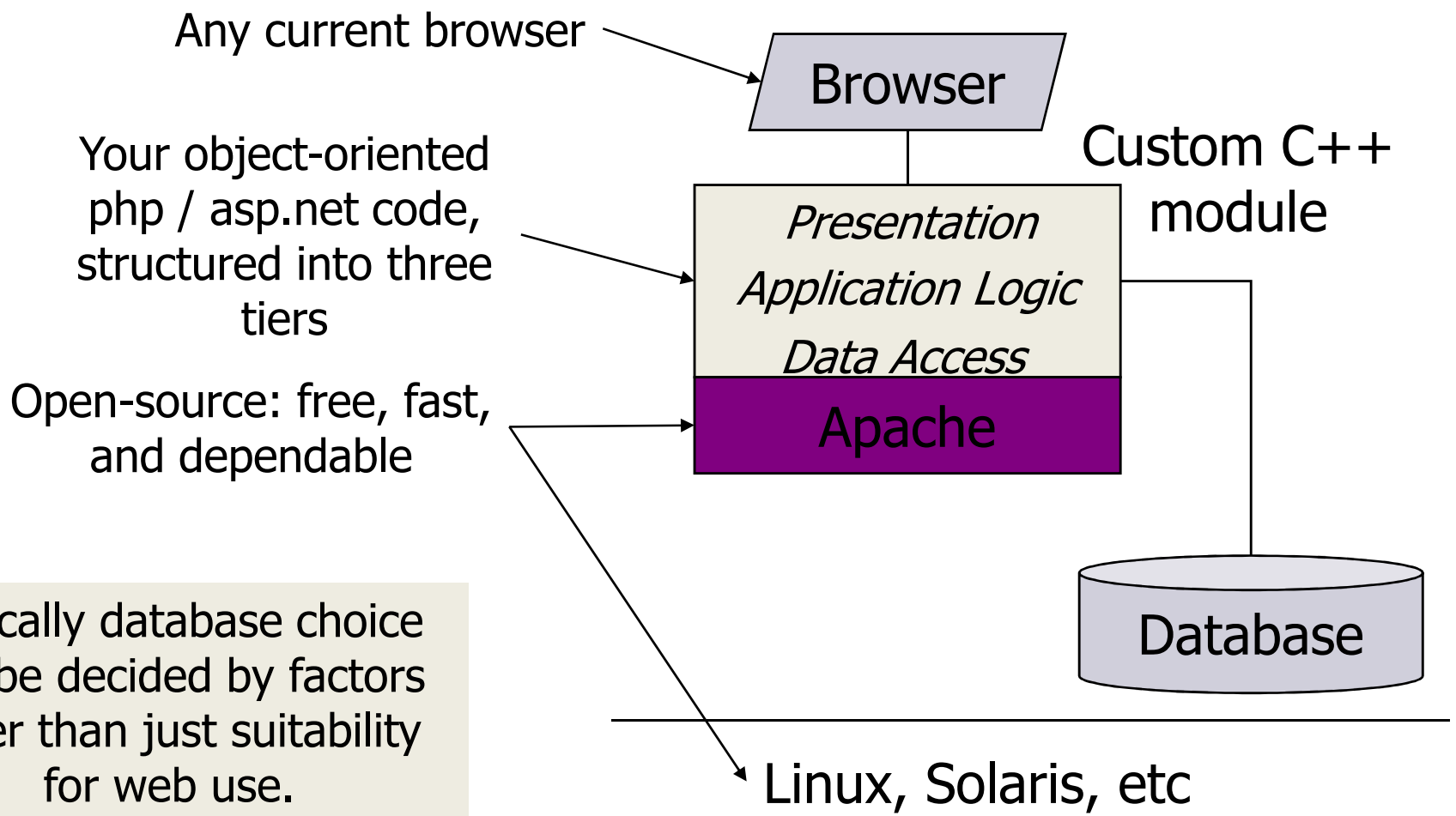
Two-Tier Architecture



LAMP: Linux, Apache, MySQL, PHP.
US$10+/month from
http://www.he.net

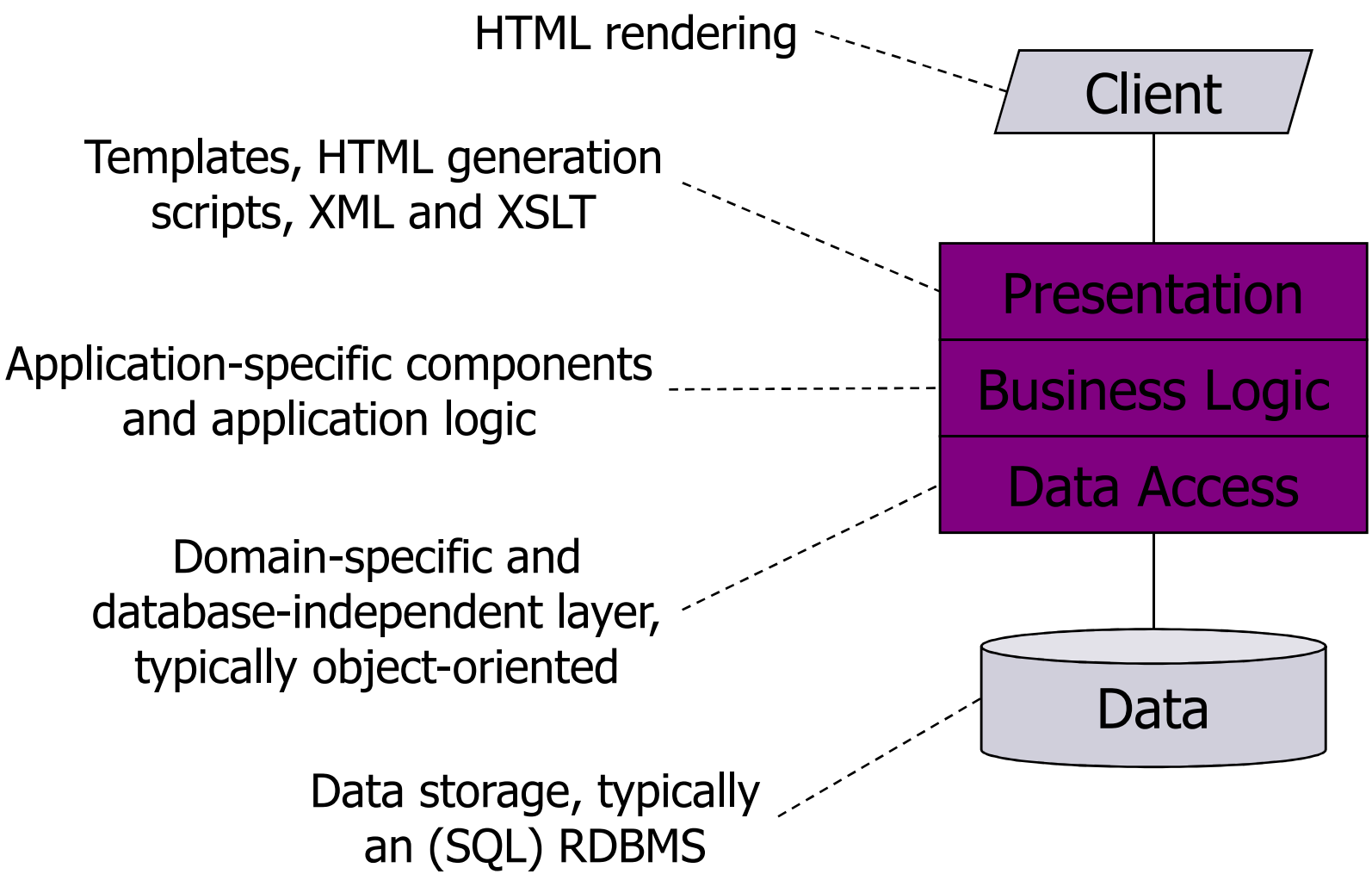
Three-Tier Architecture

...the traditional way...?



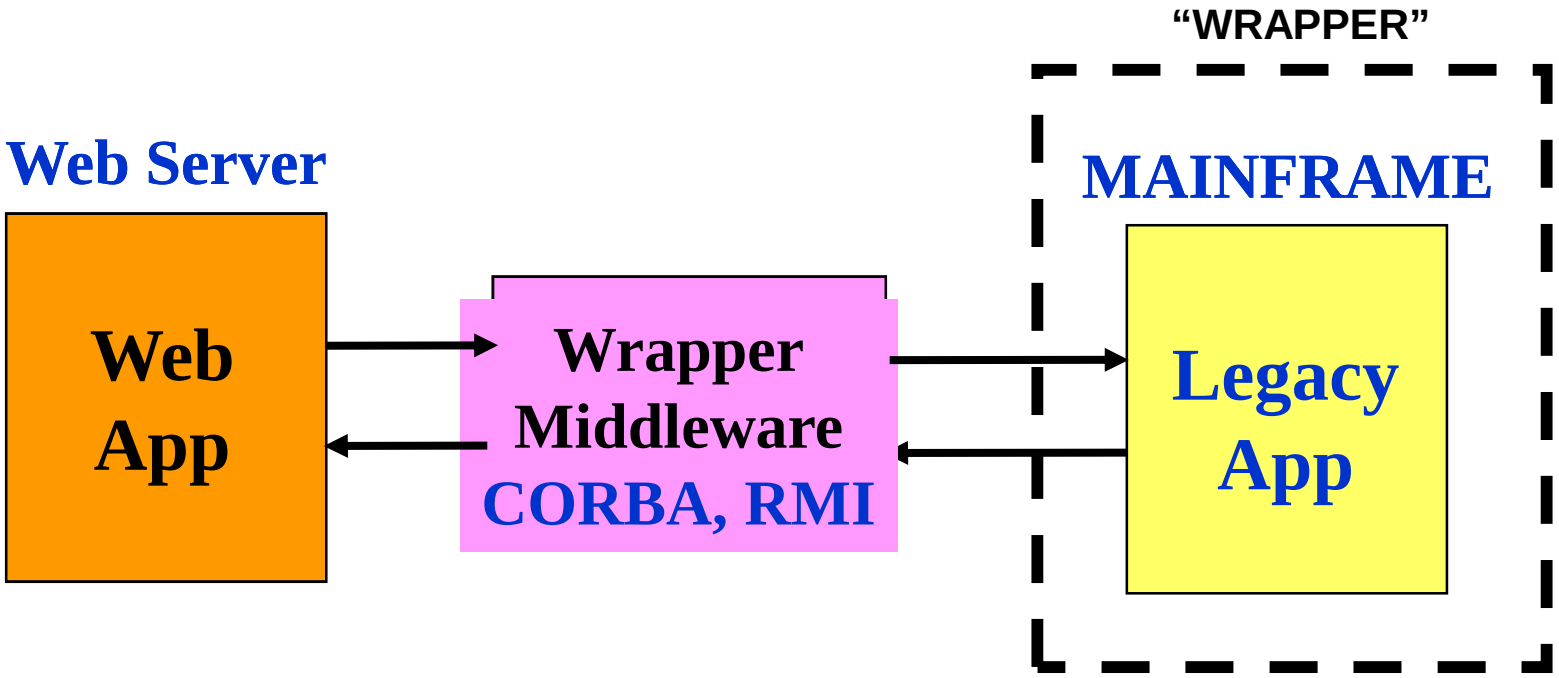
Typically database choice will be decided by factors other than just suitability for web use.

N-tier *web* architectures



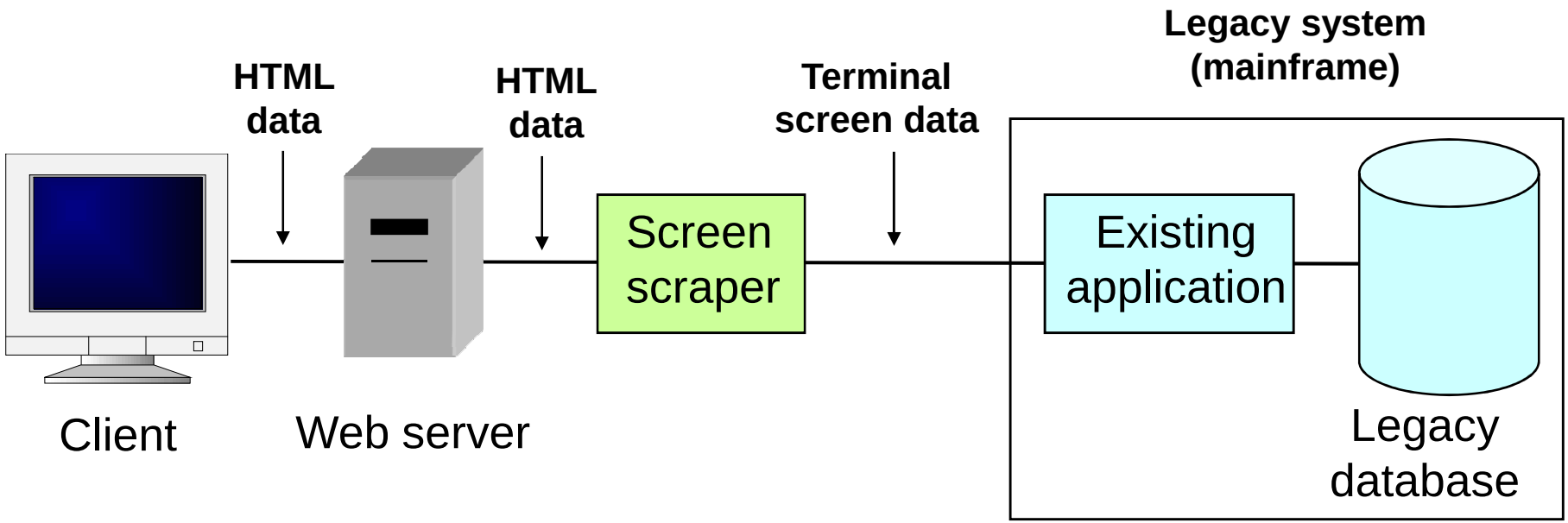
Web Integration Problem

Connecting to Legacy Systems by “Wrapping”



Connecting to Legacy Systems

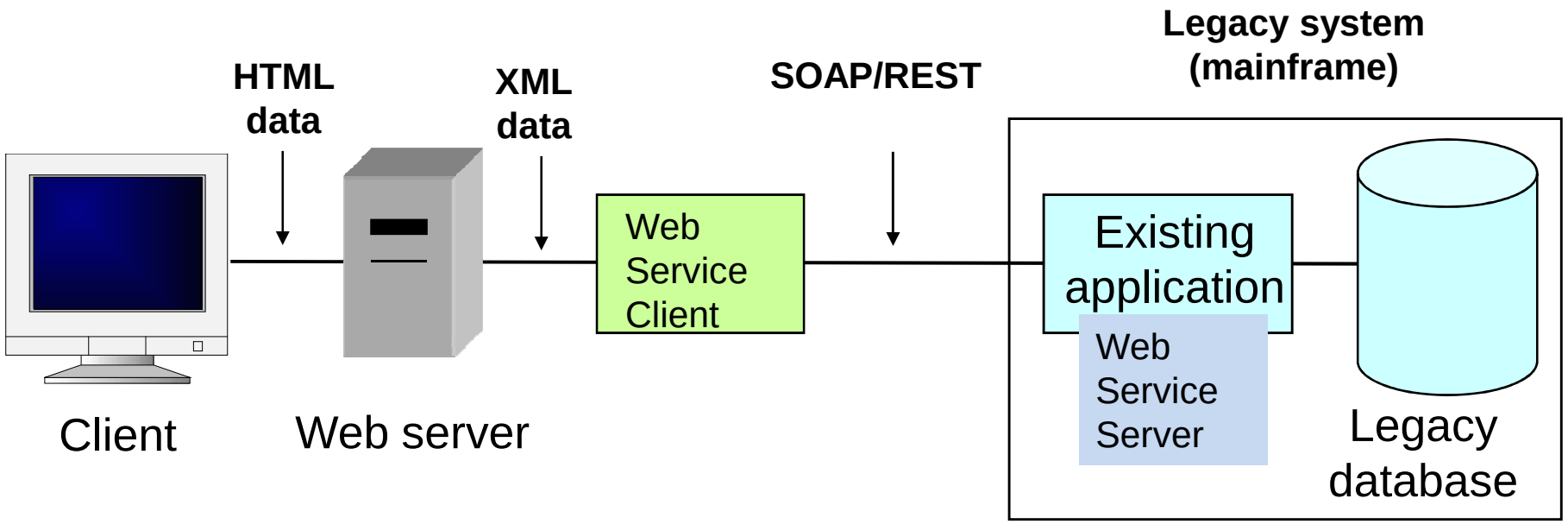
The “screen scraper”



SOURCE: WIM GEVERS

Connecting to Legacy Systems

The “web sevices”



SOURCE: WIM GEVERS

Web application architecture

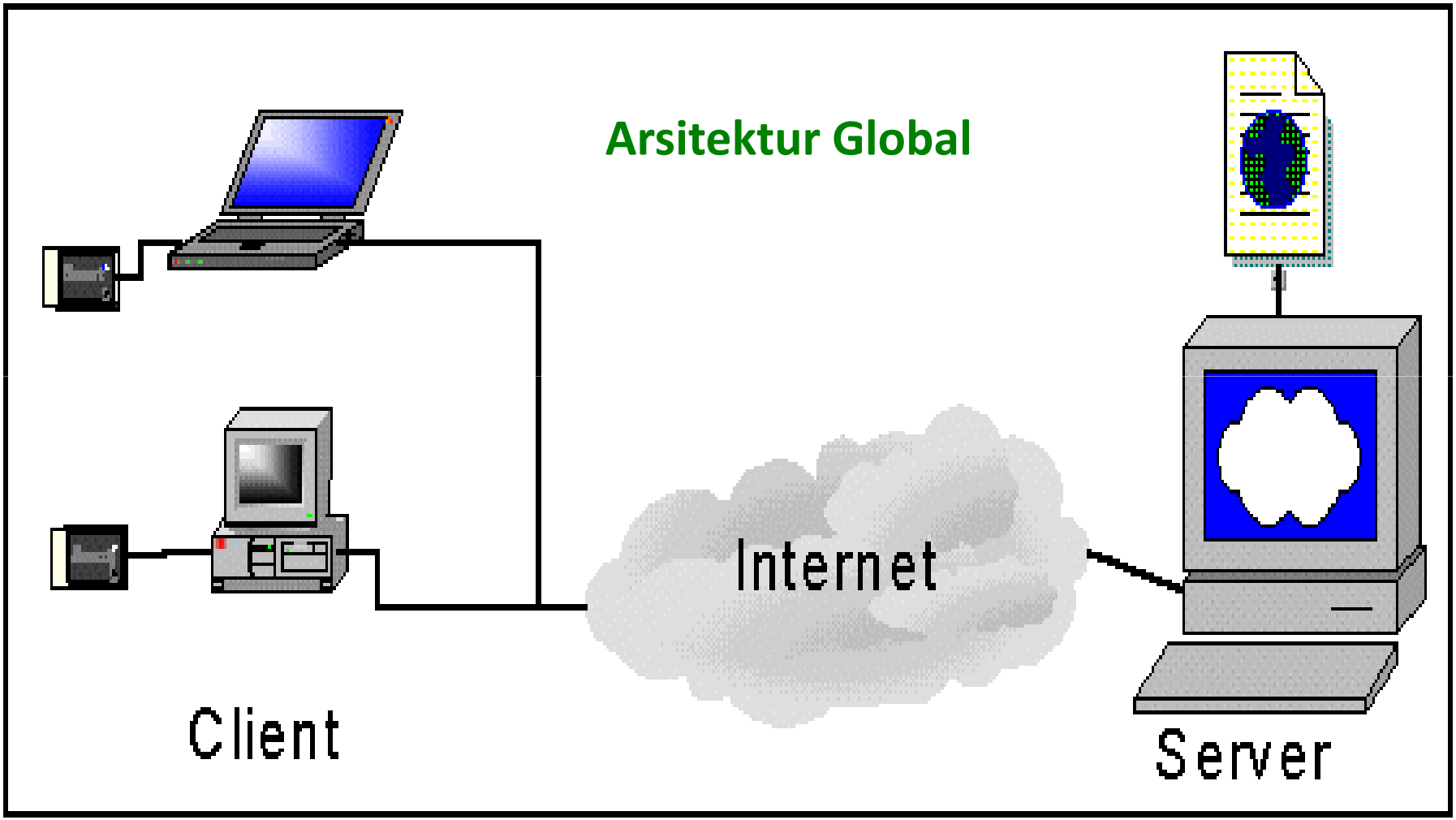
Server side

- Server hardware
- Web servers software
  - Apache, Xitami, IIS, GlassFish
- Server side programming language
  - PHP, JSP, ASP, ASP.NET, Ruby on Rails, Python, Perl, Cold Fusion
- MIME configuration
- Utility Programming Tools:
- Database Server
  - MySQL, SQLServer, Oracle

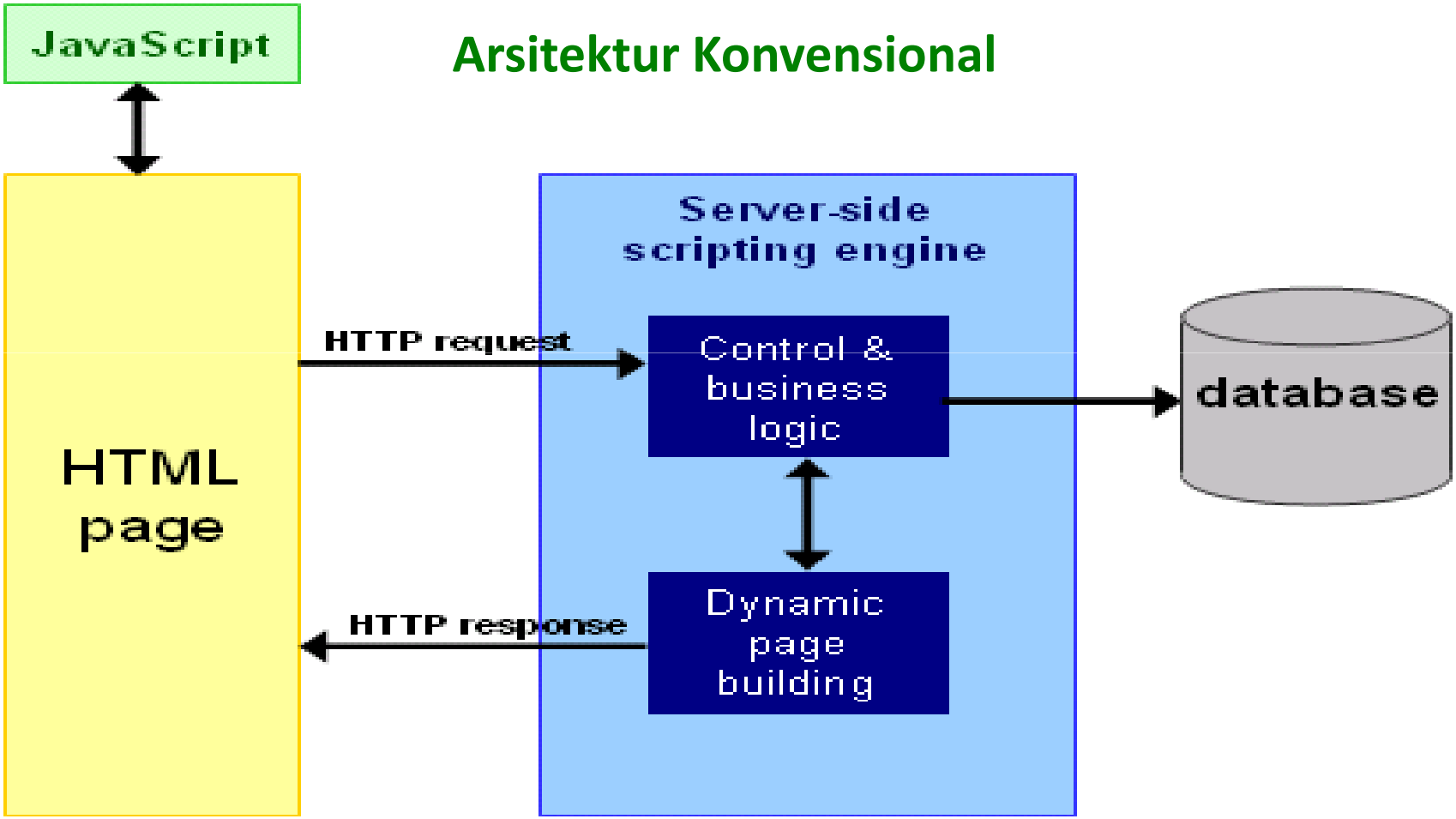
Client side

- Client hardware
- Web browsers
  - MIME config
- Language: HTML / XML / XHTML
- Client side scripting: Javascript / VBScript
- Design: CSS
- Interactive software:
  - Flash player
  - Java Applet
- Client software:
  - ActiveX / Plugin: program yg terintegrasi dgn browser
  - Helper: program yg terinstall di client

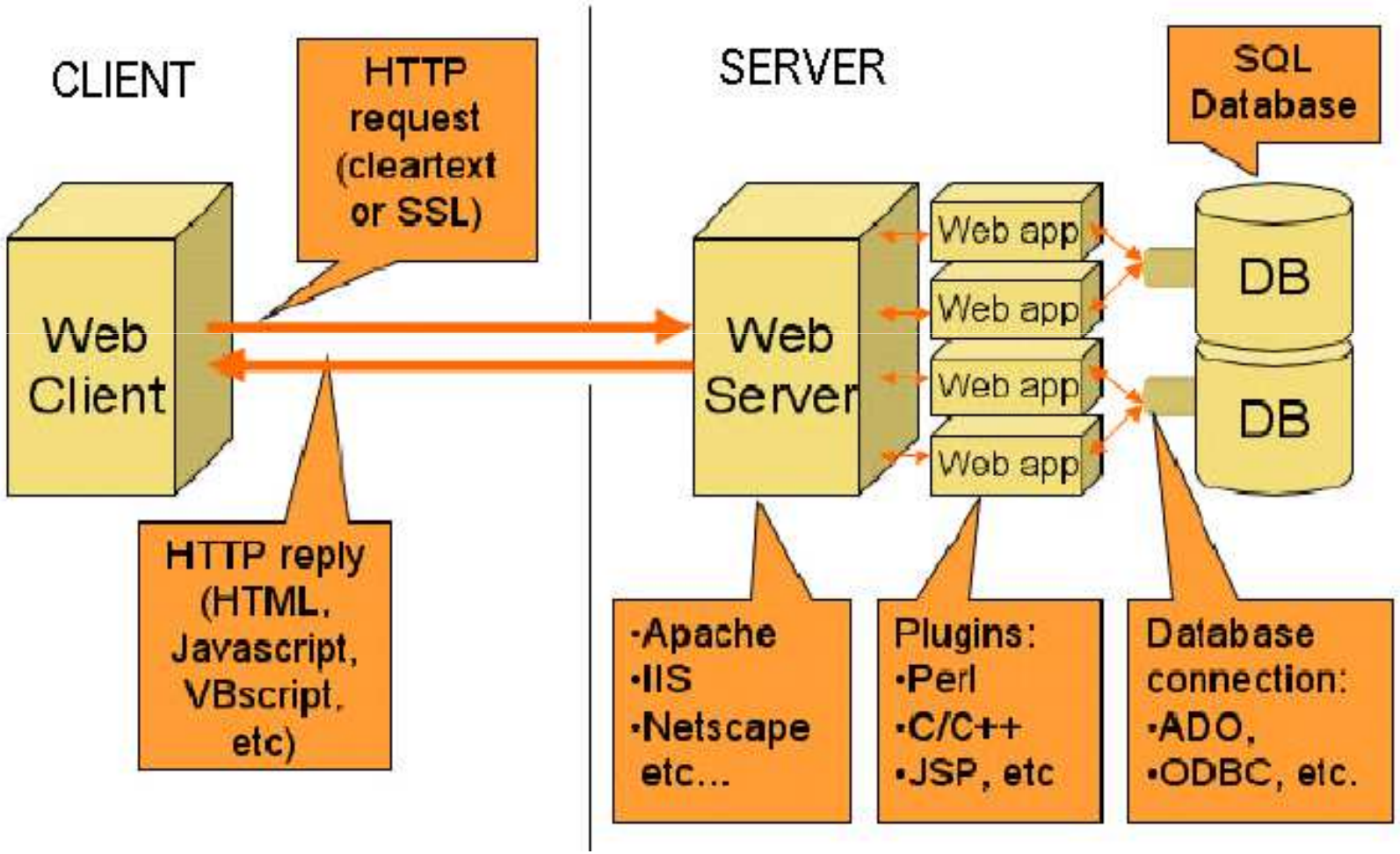
Arsitektur Aplikasi Web umum



Arsitektur Aplikasi Web konvensional

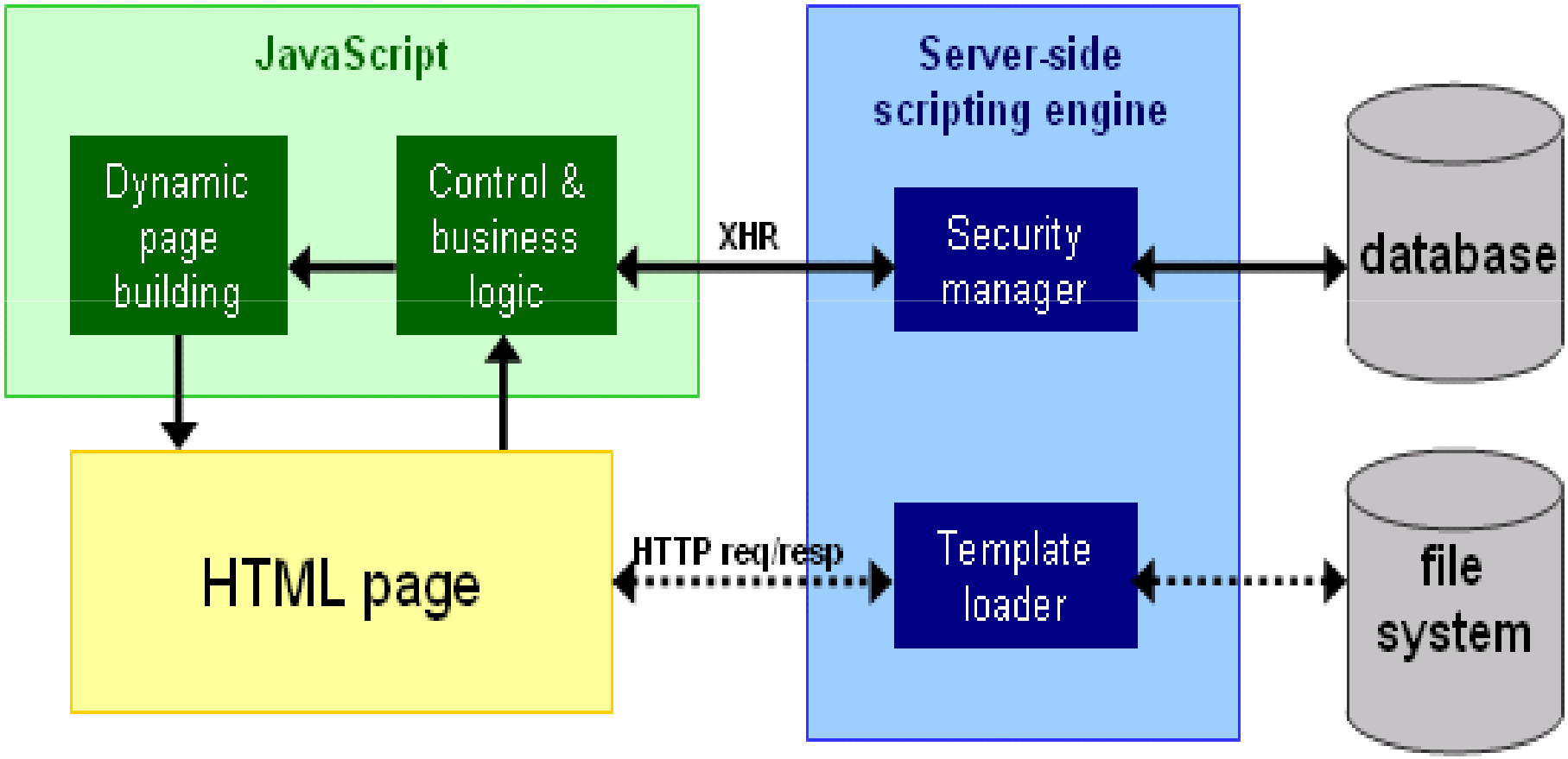


Komponen Web Detail



Arsitektur Aplikasi Web Sekarang

Tren Sekarang (Ajax-Based)



Web Data Aspect architecture

Data aspect architecture

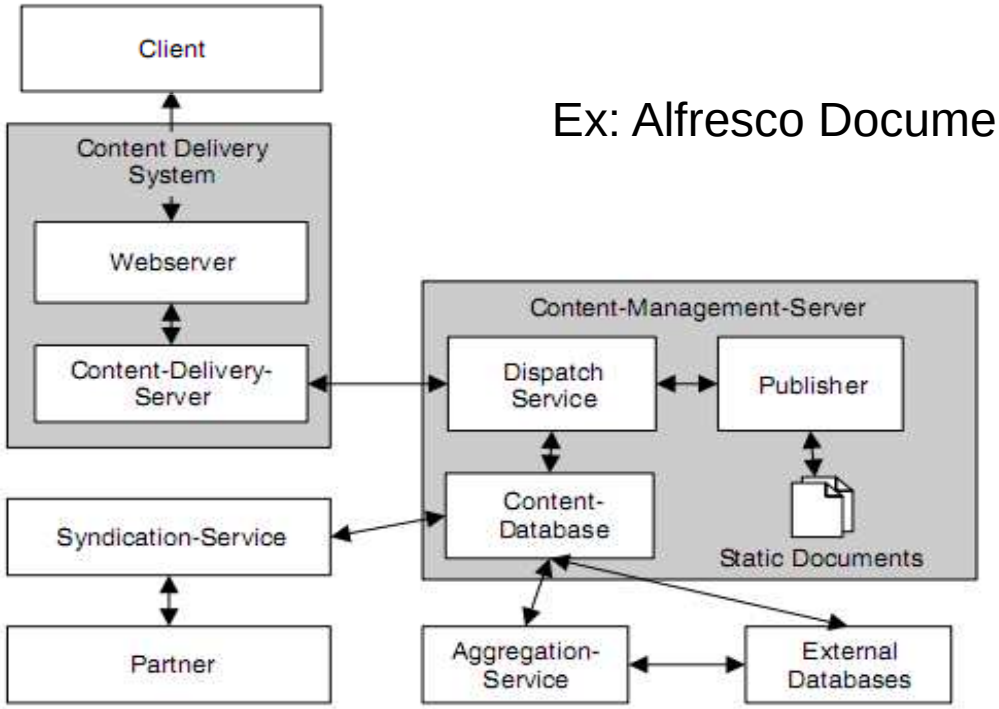
- **Structured data** of the kind held in **database**
- **Documents** of the kind used in **document management systems**
- **Multimedia data** of kind held in **media servers**

Database centric architecture

- **Integrate DB into Web app**
- DB are accessed either directly from within Web server extension or application servers
  - JDBC
  - ODBC
  - ADO
  - ADO.NET

Architecture of Web Document Management

- **Content management architecture**
  - Integrate documents from different sources
  - Representing mechanism to integrate contents into Web

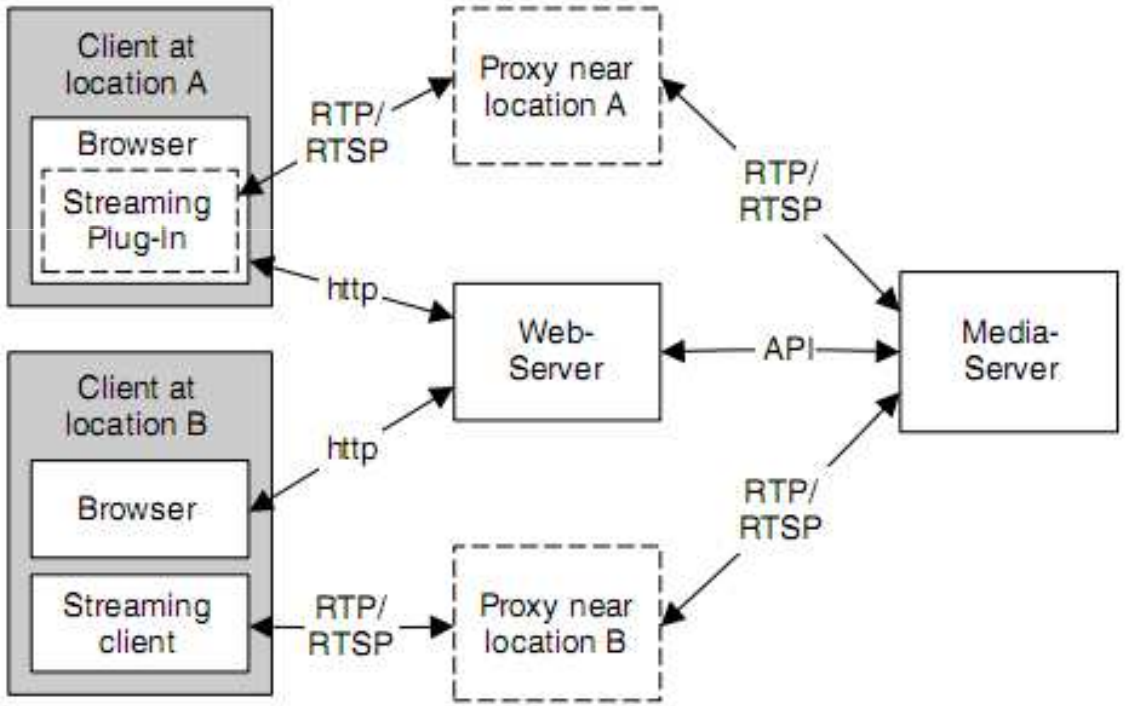


Ex: Alfresco Document Management Systems

A content management architecture for Web applications (according to Anastopoulos and

Architecture of multimedia data

- **Streaming**
  - Client can begin play out of audio few second after it begins receiving the file from server
  - Corresponding bandwidth, low jitter
  - Real time protocol
- Example: **Helix Media Server**



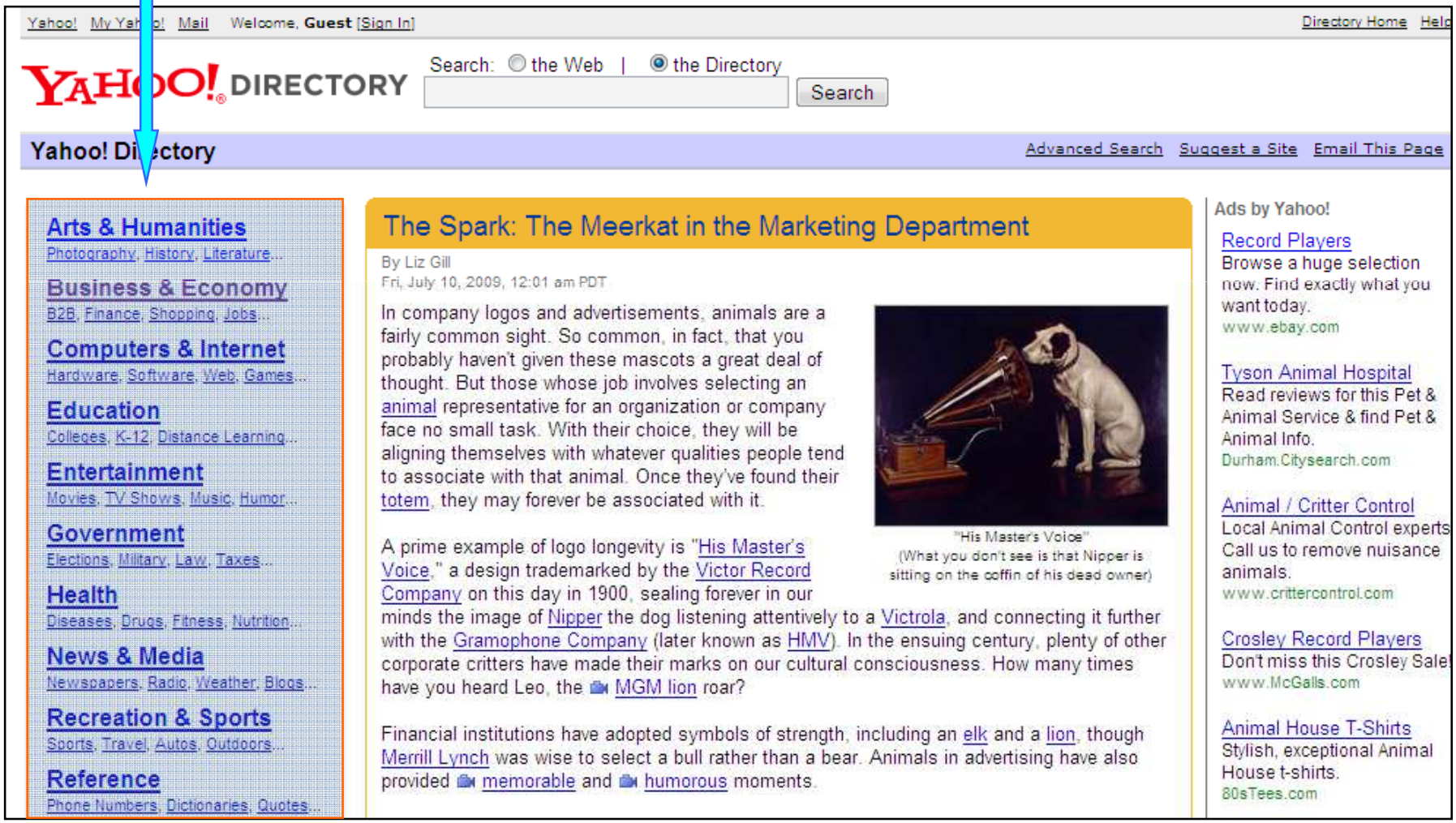
Streaming media architecture using point-to-point connections.

Now: Web Application

Modern Web App
Case Study : Facebook

Previous Web App
Case Study : Yahoo! Directory

Provider-defined directory



Examples of two “versions” of web apps

1995-2005 Web	**2005-Present Web**
Britannica Online	Wikipedia
Akamai	BitTorrent
Directories (Taxonomy)	Tagging (Folksonomy)
Tightly coupled apps	App Mashup/Integration
Home page	Blog

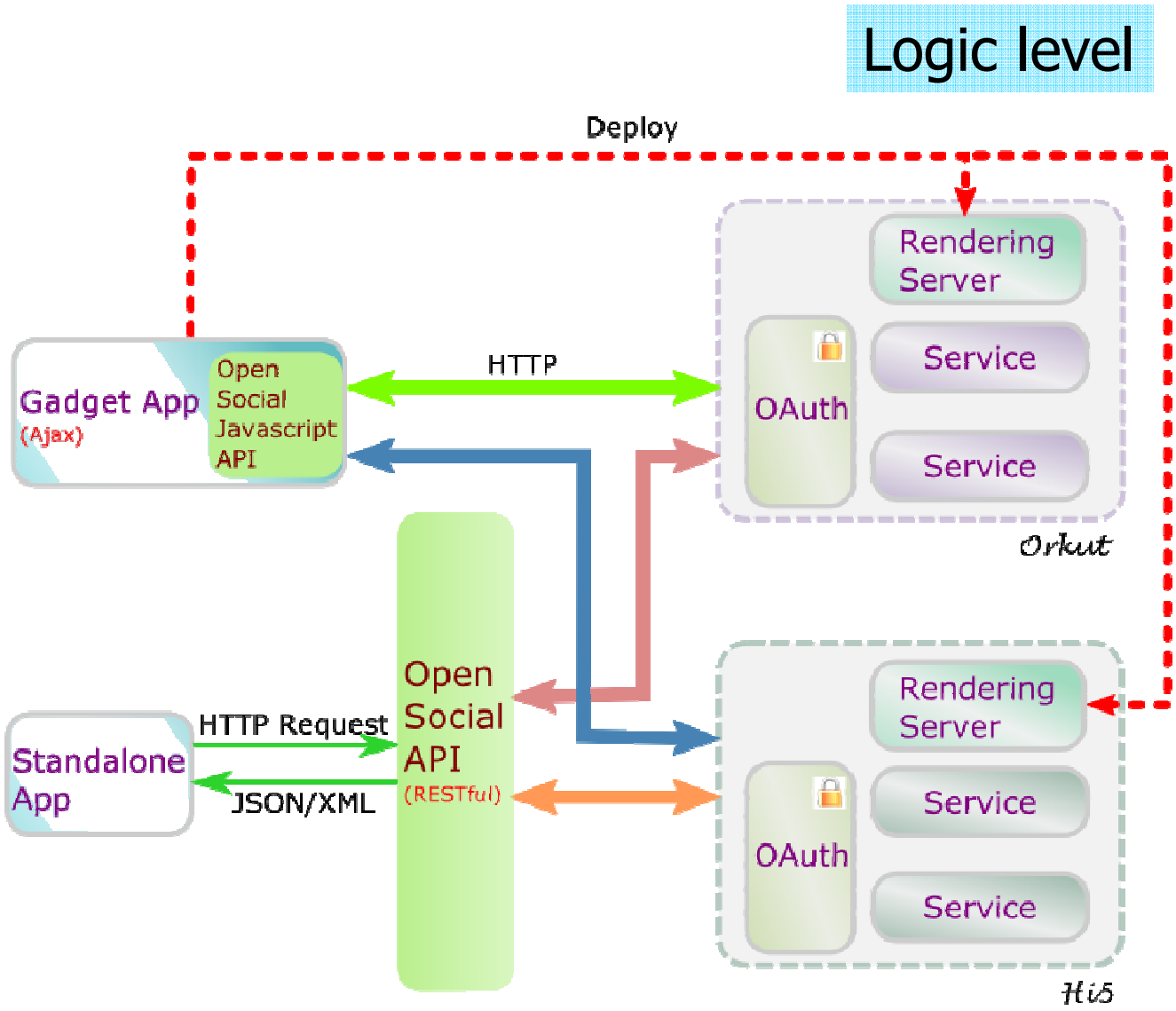
OpenSocial

- A coherent open architecture designed for **social network** services and applications.
  - Common APIs across many websites
    - REST/RPC protocols – for server-to-server interactions
    - Javascript APIs – for browser-to-server interactions
  - Authorization mechanism, Data model …
- Usage
  - Supported by MySpace, Google Orkut, Twitter, LinkedIn, XiaoNei…
    - Internationalization
  - Rival: Facebook

OpenSocial – Architecture example

Components

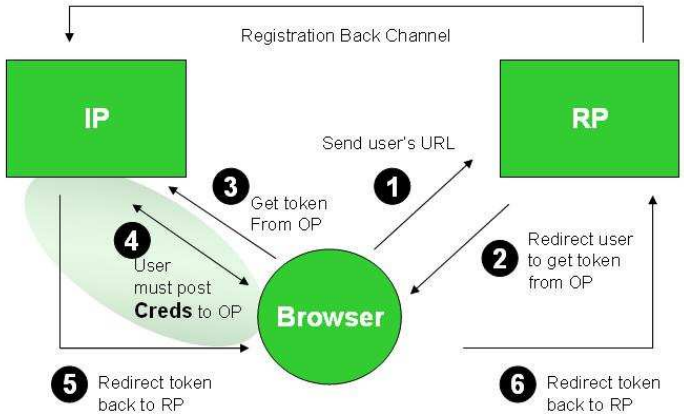
- Interface – REST, Javascript APIs
- Client – Ajax, Gadget
- Message Format – JSON, XML
- Security - OAuth
- Data Model



Authentication – OpenID

- Motivation
  - Provide lightweight **authentication** service across domains
- Solution

  Users are asked to prove ownership of their OpenID identifiers.
  - OpenID identifiers are URLs (e.g. http://zhenhua-guo.blogspot.com).
  - Service provider and identity provider are clearly separated.
  - Authentication delegation (service provider → identity provider)
- Advantages
  - Cross-domain authentication
  - Attribute exchange beyond authentication
  - Single Sign-On
  - Easy OpenID provider switch
- Drawbacks
  - Phishing attack
- Resources
  - Supported by Facebook, Verisign, Sourceforge, Yahoo, etc.



http://fcom.us.es/blogs/nuevafcom/files/2008/09/openid-1.jpg

Next

- Service Oriented Architectures