HTTP (Web) server

Henry Saptono

2013/2014

WWW (W3)

- World Wide Web (disingkat sebagai WWW atau W3, umumnya dikenal sebagai web) adalah sistem dokumen *hypertext* yang diakses melalui Internet.
- Dengan web browser, kita dapat melihat halaman web yang mungkin berisi teks, gambar, video, dan multimedia lainnya dan menavigasi antara mereka melalui *hyperlink*

Sejarah WWW

World Wide Web dimulai sebagai proyek CERN bernama Enquire, diprakarsai oleh Tim Berners–Lee pada tahun 1989 dan Robert Cailliau pada tahun 1990. Berners–Lee dan Cailliau diberi penghargaan oleh Association for Computing Machinery pada tahun 1995, Atas kontribusi mereka kepada pengembangan World Wide Web.

Berdasarkan konsep hypertext, proyek ini bertujuan untuk memfasilitasi berbagi informasi di antara para peneliti. Situs web pertama di on–line kan pada tahun 1991. Pada tanggal 30 April 1993, CERN mengumumkan bahwa World Wide Web akan bebas untuk siapa pun. Salinan asli halaman pertama web, yang diciptakan oleh Berners–Lee, masih dipublikasikan di situs World Wide Web Consortium sebagai dokumen sejarah.

Sebelum pengembangan Web, CERN telah menjadi pelopor dalam pengenalan teknologi internet, dimulai pada awal tahun 1980. Sejarah singkat periode ini dapat ditemukan di CERN.ch.

Web site

- *web site* atau *web* (situs web) adalah kumpulan file-file *world wide web* (www) yang berisi file permulaan yang disebut halaman rumah (*home page*).
- Sebuah institusi atau perusahaan memberitahu Anda bagaimana untuk sampai ke situs web mereka dengan memberikan alamat halaman rumah mereka (URL). Dari halaman rumah, Anda bisa mendapatkan halaman lain di situs web mereka.

Contoh:

situs web untuk STT Terpadu Nurul Fikri memiliki alamat halaman rumah http://www.nurulfikri.ac.id. Alamat halaman rumah STT Terpadu Nurul Fikri akan menghantarkan pengunjung web STT NF ke puluhan atau mungkin ratusan halaman lainnya

Konsep kerja web

- Web bekerja dalam konsep *client – server*
  - Web client meminta halaman (dokumen web) kepada web server melalui jaringan komputer
  - Web server merespon dengan mencarikan dokumen (halaman) pada lokasi penyimpanan dokumen-dokumen web yang ada pada web server
  - Selanjutnya web server mengirimkannya kepada web client
- Contoh aplikasi web client: Mozilla Firefox, IE, Opera, Safari, Chrome dan lain lain.
- Coontoh aplikasi web server: Apache HTTP server, IIS, Nginx, lighttpd, tomcat dan lain lain.

Web Site

Sederhana, seluruh solusi dalam server tunggal

Web client

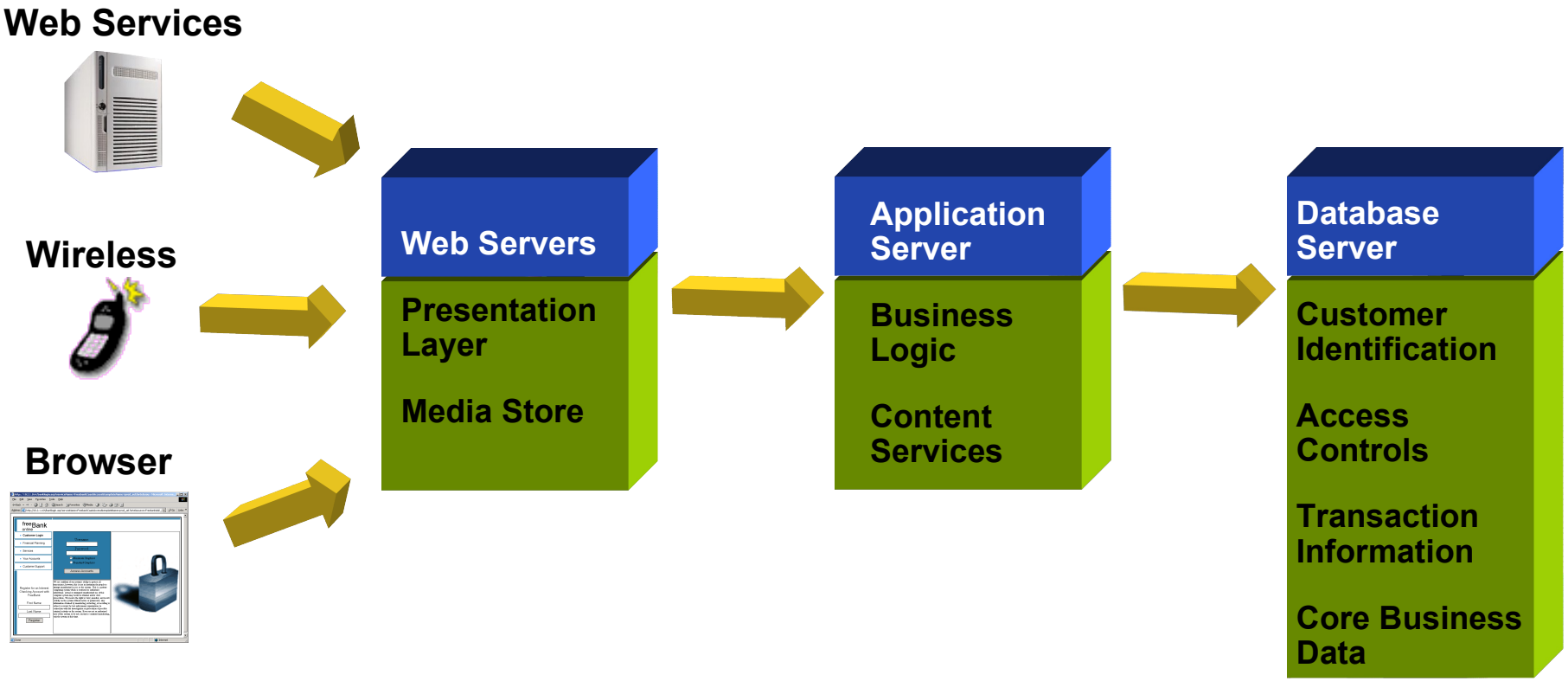
(Browser)

Web Server

HTML

Web Applications

Arsitekturnya kompleks, multiple platforms, multiple protocols



Seperti apa dokumen web ?

- Sebenarnya yang terjadi antara web client dan web server adalah proses transfer file file dokumen web
- Agar mudah dipahami atau dimengerti oleh web client maka dokumen web harus memiliki format yang standar
- Format dokumen web secara *de facto* adalah menggunakan format bahasa HTML (*HyperText Mark-up Language*)
- File format HTML ini biasanya berekstensi .html atau .htm

Contoh format HTML

```
<HTML>
<HEAD>
   <TITLE>Hello world</TITLE>
</HEAD>
<BODY>
<DIV> Hello World </DIV>
<DIV><H3>Selamat datang di web saya</H3></DIV>
</BODY>
</HTML>
```

Bahasa pemrograman web

- Dalam perkembangannya, web tidak lagi hanya sekedar menampilkan informasi yang bersifat statis, melainkan juga dinamis dengan unsur unsur logika melibatkan basis data (*web applications*)
- Untuk itu dibutuhkan mekanisme untuk menghasilkan file berformat HTML saat proses transfer file (*on the fly*) dari web server ke web client. Mekanisme itu terjadi di sisi web server yang ditangani oleh sebuah interpreter bahasa pemrograman.
- Bahasa pemrograman yang menginterpretasi kode program menjadi file atau *stream data* berformat HTML yang ditransfer ke web client oleh web server disebut ***bahasa pemrograman web (server side language)***

Protokol web

- Untuk berkomunikasi dalam proses transfer dokumen web, antara web client dan web server, dibutuhkan serangkaian prosedur dan aturan dalam berkomunikasi yang disepakati dan dimengerti oleh keduanya.
- Prosedur dan aturan berkomunikasi tersebut dinamakan protokol. Maka prosedur dan aturan berkomunikasi antara web client dan web server kita sebut dengan istilah **Protokol Web**

Protokol HTTP dan HTTPS

Protokol web terdiri dari dua protokol yaitu:

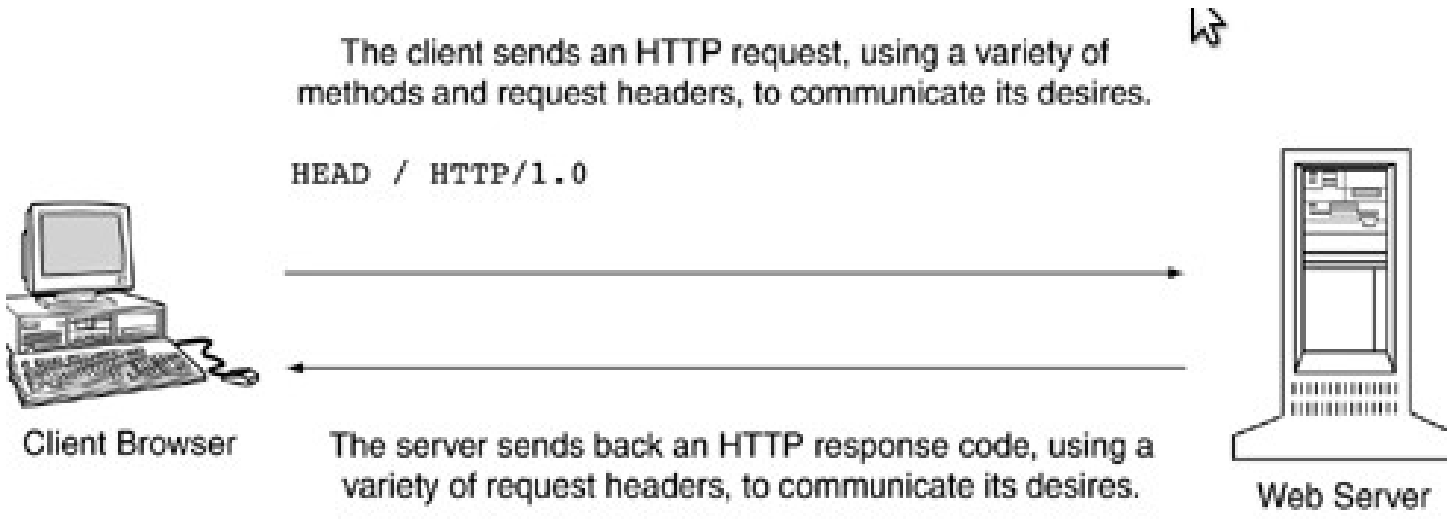
- HTTP (HyperText Transfer Protocol), ini adalah protokol default komunikasi web, protokol ini secara lahiriah membawa kerentanan yang dikarenakan proses transfer data berupa clear text data. Nomor port service 80.
- HTTPS (HyperText Transfer Protocol Secure), ini disebut juga dengan istilah HTTP over SSL. Protokol ini relatif lebih aman dikarenakan telah menerapkan proses enkripsi data yang akan ditransfer. Nomor port service 443.

HTTP

Setiap web client (browser) dan server harus menggunakan protokol HTTP ini untuk berkomunikasi.

HTTP adalah suatu protokol *request/response stateless*, yang memungkinkan komputer komputer berkomunikasi satu dengan yang lainnya lebih efisien, dan dapat melakukan komunikasi yang berlangsung lama.

Ilustrasi protokol HTTP




```
HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Content-Location: http://192.168.0.5/
Default.htm
Date: Mon, 15 Apr 2002 05:56:57 GMT
Content-Type: text/html
Accept-Ranges: bytes
Last-Modified: Sat, 06 Apr 2002 06:48:32 GMT
Etag: "a017751137ddc11:81a"
Content-Lenght: 5
```

HTTP Request

Request Method	Syntax and Notes
CONNECT	CONNECT proxy-server HTTP/1.1 Host: server Proxy-Authorization: basic dWNpOjIwMDM= *Set up a tunnel through proxies. The "Proxy-Authorization" header is present only if authentication is required.*
DELETE	DELETE /uri HTTP/1.1 Host: website *Delete the resource from the server.*
GET	GET /uri HTTP/1.0 *Retrieve the information associated with "/uri".*
HEAD	HEAD /uri HTTP/1.0 *Identical to GET, but the server does not return the message body of the resource. In other words, the server only supplies the HTTP status code and relevant headers.*

HTTP Request

Request Method	Syntax and Notes
POST	POST /uri HTTP/1.1 Host: website Content-Length: N \n \n <post data> *Instruct the server to accept "<post data>" to the requested resource. The POST will define the content-length, content-type, and may contain binary data. Originally, this was intended to append "<post data>" to the resource.*
PUT	PUT /uri HTTP/1.1 Host: website Content-Length: N \n \n <put data> *Instruct the server to place "<put data>" in the location designated by the URI.*
TRACE	TRACE / HTTP/1.1 Host: website *Cause the server to respond with all of the headers specified in the original request.*

HTTP Response

Suatu HTTP Request dari web client ditangani oleh server dan direspon. Pada saat server merespon, server mengirimkan kembali serangkain komponen-komponen pesan yang dapat dikategorikan sebagai berikut:

- **Response code** – adalah kode bilangan yang berhubungan dengan respon terkait.
- **Header fields** – informasi tambahan mengenai respon
- **Data** – isi dari respon

HTTP Response code

Response Code	Description
Success 2xx	
200 OK	The request has succeeded.
Redirection 3xx	
301 Moved Permanently	The requested resource has been given a new permanent URL, which will be placed in the Location field. This response code is saying, "I have moved, follow me to my new home."
302 Moved Temporarily	The requested resource has been given a new temporary URL, which will be placed in the Location field. This response code is saying, "I have moved, follow me to my temporary home, but don't depend on me being here long."
Client Error 4xx	
400 Bad Request	The request wasn't understood by the server.
401 Unauthorized	The resource requested requires user authentication, usually in the form of Basic or equivalent authentication.
403 Forbidden	The server understood the request but is refusing to respond. Typically, when the GET method is used to receive this response, little or no further information will be present. However, when the HEAD method is used, some servers will give more detailed information about why this condition occurred.
404 Not Found	The requested resource wasn't found.
Server Errors 5xx	
500 Internal Server Error	The server discovered an internal error in processing the request.
501 Not Implemented	The server doesn't support the request.
502 Bad Gateway	The server received an invalid response from an upstream server when it requested the desired resource. This response is typical of HTTP proxies.
503 Service Unavailable	The server is unable to respond to the request because it is being overwhelmed.

Header Fields definitions

Header Field	*Description*
Allow	Lists the methods supported by the resource requested.
Authorization	Lists the authorization credentials for HTTP authentication.
Content-Encoding	Lists any additional content encoding being performed on the data returned. With this information the client knows better how to interpret the data returned. For example, Content-Encoding: x-gzip means that the content is gzip compressed.
Content-Length	Lists the size of the content's body in decimal number of octets. For example, `Content-Length: 332`.
Content-Type	Lists the content's type in the response. For example, `Content-Type: text/html` lists `text/html` as the content's type. This field helps the client understand better how to display the content in the browser.
Date	Lists the server's date and time.
Expires	Lists the date and time that the content should be considered out of date.
From	Lists an e-mail address to be used for identifying the content's responsible party. This field is rarely used.
Last-Modified	Lists the date and time that the server believes the requested resource was last modified.
Location	Lists the location of the resource requested.
Pragma	Describes optional behavior for requests. For example, if the Pragma header field is sent from the server with the "no-cache" directive, the client should load the content sent regardless of whether it has cached a copy of it, as in `Date: Thu, 22 Apr 2002 01:10:22 GMT`.
Referer	Allows the client to specify the address of the resource.
Server	Lists the software running on the server. In most cases, this information is accurate. For example, `Server: Microsoft-IIS/5.0`. However, be forewarned because some smart administrators can change this information to be anything they want, such as "Mickey's Web Server."
User-Agent	Lists additional information about the user agent (client) requesting information. For example, `User-Agent: Mozilla/5.0 (WinNT)`.
WWW-Authenticate	Used in response to 401 Unauthorized response code, this field holds a challenge for negotiating with the server for authorized access.

Mengamati traffic HTTP

Cara mudah kita mengamati *http request method* dan *http response header* adalah dengan menggunakan perangkat lunak ***tcpdump***

Berikut ini contoh penggunaan tcpdump untuk mengamati traffic http dengan alamat web www.nurulfikri.ac.id

```
# tcpdump -i eth0 -A -s 1492 'dst port 80 and dst
  host www.nurulfikri.ac.id'
```

HTTPS (*HTTP over SSL*)

- HTTPS adalah protokol yang digunakan untuk lalu lintas yang dienkripsi dalam *stream* HTTP.
- Seluruh pesan akan dienkripsi jika *Secure Socket Layer* (SSL) digunakan.
- Banyak versi SSL dan protokol yang terkait (Transport Layer Security, TLS, dan RFC2246) yang tersedia, termasuk SSLv1, SSLv2, dan SSLv3.
- Dan untuk membuat lebih menyulitkan, SSL menawarkan berbagai pilihan untuk standar enkripsi tertentu yang digunakan dalam SSL. Misalnya, dengan SSLv3, Anda dapat memilih dari DES ke RSA (RC2 dan RC4).

Mengamati traffic HTTPS

Untuk mengamati traffic https gunakan perangkat tcpdump, seperti contoh berikut ini:

 # tcpdump -i eth0 -A -s 1492 'dst port 443 and dst host www.nurulfikri.ac.id'

Karena pada HTTPS pesan yang dilewatkan dienkripsi maka Anda tidak melihat pesan clear text sebagaimana yang Anda lihat pada HTTP

SSL mengenkripsi lalu lintas antara server dan client, ini secara signifikan mengurangi penyerang mengakses informasi sensitif (misal:password) dan merekamnya.

URL

- URL atau Uniform Resource Locator
- Secara umum, URL adalah mekanisme untuk mengidentifikasi sumber daya diinternet (*web service* dan *ftp service*), yang membuat permintaan ke suatu penyedia service (server).

- Struktur generik dari URL:

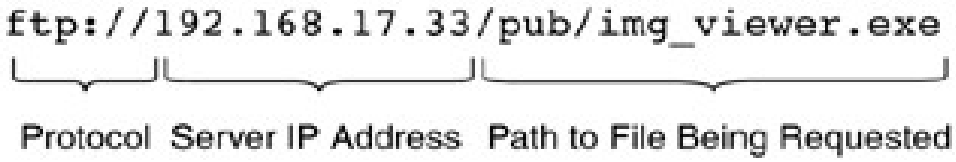
protocol://server/path/to/resource?parameters

Komponen URL

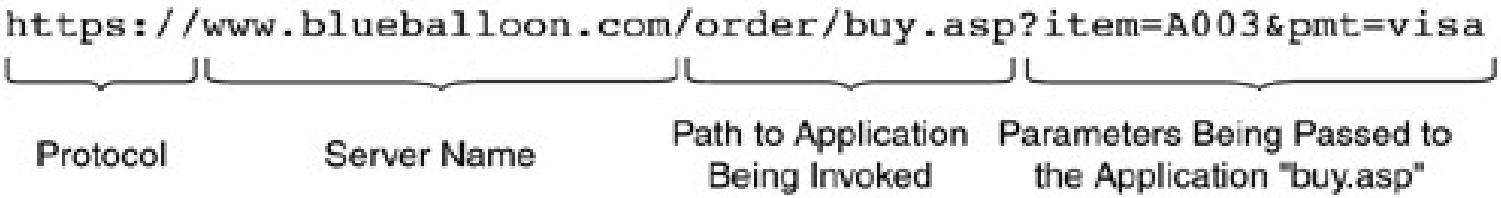
- **Protocol** – *Application layer protocol*, adalah komponen URL yang digunakan untuk meminta sumber daya dari web server, biasanya adalah protocol http:, atau protocol lainnya seperti ftp:, https:, ldap: , tergantung pada dukungan browser dan server.
- **Server** – Nama host penyedia sumber daya (nama web site), biasanya nama yang terdaftar pada DNS, atau berupa alamat ip address.
- **path/to/resource** – Path direktori, meliputi nama sumber daya yang diminta oleh browser. Sumber daya bisa berupa file statik, atau sebuah aplikasi yang menghasilkan output dinamik.
- **Parameters –** sesuatu yang opsional. Parameter dapat dilewatkan ke sumber daya, jika sumber daya adalah suatu aplikasi atau program yang menghasilkan output dinamik. Kadang *parameters* disebut juga sebagai *query string*.

Contoh URL

(a)



(b)



(c)

HTML Form

Hampir semua aplikasi web interaktif menggunakan form HTML dalam satu atau lain cara. Jika Anda menggunakan Internet, kemungkinan bahwa Anda telah berhadapan dengan HTML Form di hampir semua web ketika Anda berselancar web – seperti ketika menggunakan mesin pencari, memeriksa e-mail berbasis web, memeriksa pernyataan kartu kredit Anda, mengirimkan kartu ucapan elektronik, dan banyak tugas lainnya. HTML Form adalah antarmuka yang digunakan oleh aplikasi web untuk berinteraksi dengan pengguna melalui web browser(web client).

HTML Form

Ada dua aspek kunci dari HTML Form: penanganan disisi browser dan pemrosesan di sisi server. Saat menampilkan formulir (*form*), tugas browser terdiri dari memastikan bahwa berbagai elemen masukan (*input*) HTML akan ditampilkan dengan benar dan bahwa pengguna diijinkan untuk mengisi elemen masukan HTML dengan data dan dengan cara yang tepat. Setelah pengguna mengisi data, dia diperbolehkan untuk menyerahkan formulir yang telah diisi. Browser memastikan bahwa data dalam form dikodekan dengan benar, menggunakan URL Encoding, dan pada gilirannya akan disampaikan kepada program aplikasi yang dimaksudkan untuk menerima data dalam Form tersebut.

HTML Form

- Setelah data tersebut diterima oleh program aplikasi, berbagai elemen masukan harus dipisahkan dan diproses.
- Mulanya, diserahkan kepada programmer aplikasi untuk memasukkan logika yang diperlukan untuk mengurai data URL yang dikodekan dan mengekstraksi nama elemen input dan nilai-nilai yang disampaikan.
- Pada web server dan platform modern telah memiliki rutin *built-in* untuk memproses data formulir secara otomatis pada saat disubmit, yang tentunya menyederhanakan tugas programmer aplikasi web.

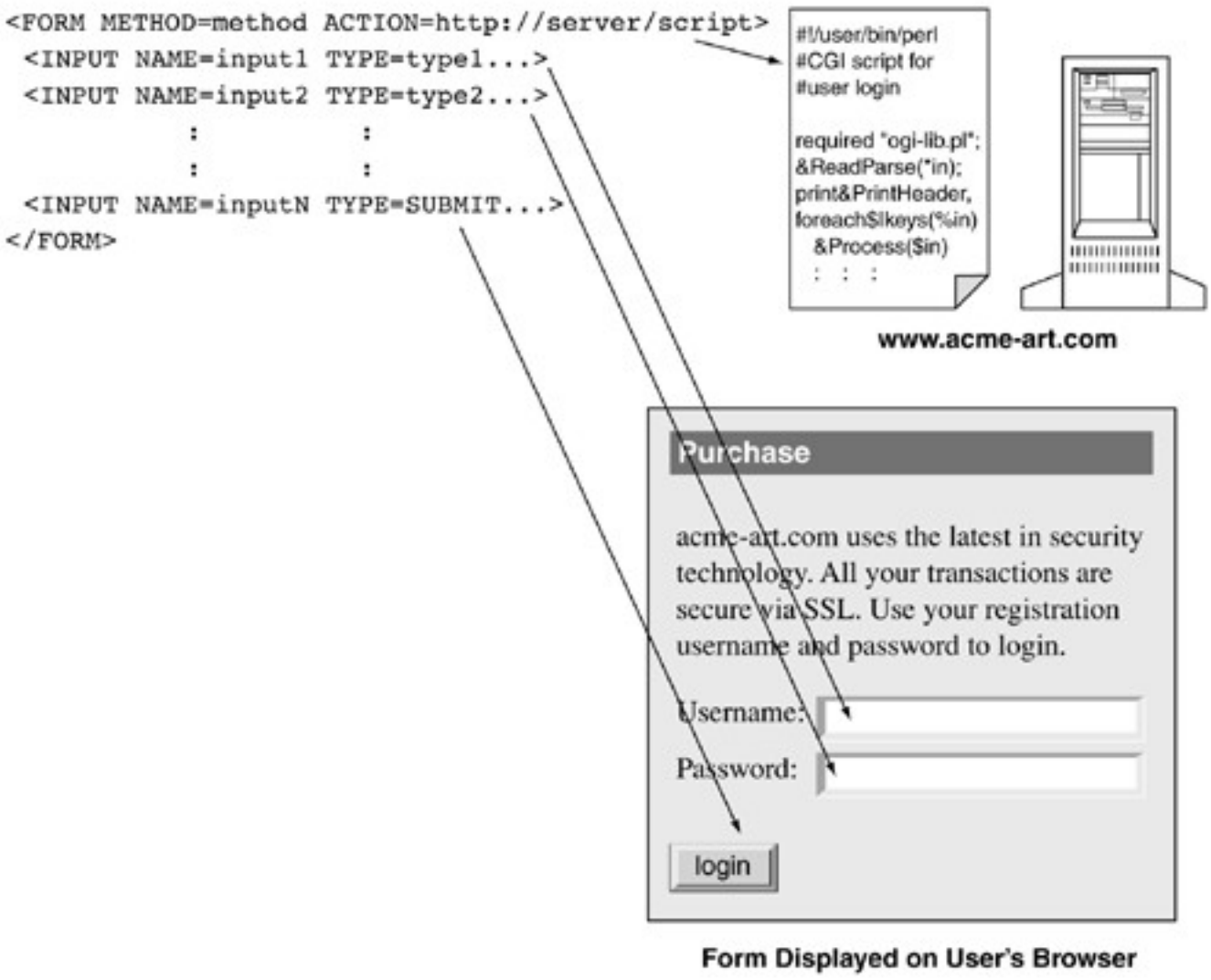
HTML Form

- Dari sudut pandang keamanan, programmer aplikasi web perlu berhati-hati terhadap masukan-masukan (*input*) yang diperbolehkan oleh browser.

Anatomi HTML Form

HTML Form diidentifikasi dengan penanda (*tag*) <FORM> ... </ FORM>. Semua tag-tag HTML yang tertanam dalam tag Form diperlakukan sebagai bagian dari formulir. Di antara tag HTML lainnya, tag <INPUT> meliptui unsur masukan dari formulir. Mereka memungkinkan pengguna untuk memasukkan data dalam HTML Form yang ditampilkan pada browser.

Anatomi HTML Form



Konsep HTML Form

Ada beberapa konsep kunci tentang HTML Form.

- **Method**: Setiap Form harus memiliki satu metode pengiriman formulir, baik GET atau POST. Ini menentukan metode HTTP yang mana yang browser gunakan saat mengirim data formulir ke server.
- **Action**: Setiap Form harus memiliki aplikasi *server-side* yang terkait. Aplikasi harus dirancang untuk menerima data dari berbagai elemen input dari suatu form.

Konsep HTML Form

- **Inputs element**: Setiap elemen input harus memiliki nama, yang digunakan oleh aplikasi server-side untuk memparsing parameter dan nilai-nilainya.
- **Submit button**: Setiap Form harus memiliki tombol Submit, yang merupakan tipe khusus dari elemen input, ditampilkan sebagai tombol yang dapat diklik. Ketika tombol Submit diklik, browser mengumpulkan dan mengkodekan data yang disediakan pengguna dari berbagai kolom formulir dan mengirimkannya ke aplikasi *server-side*

Contoh kode sumber HTML Form

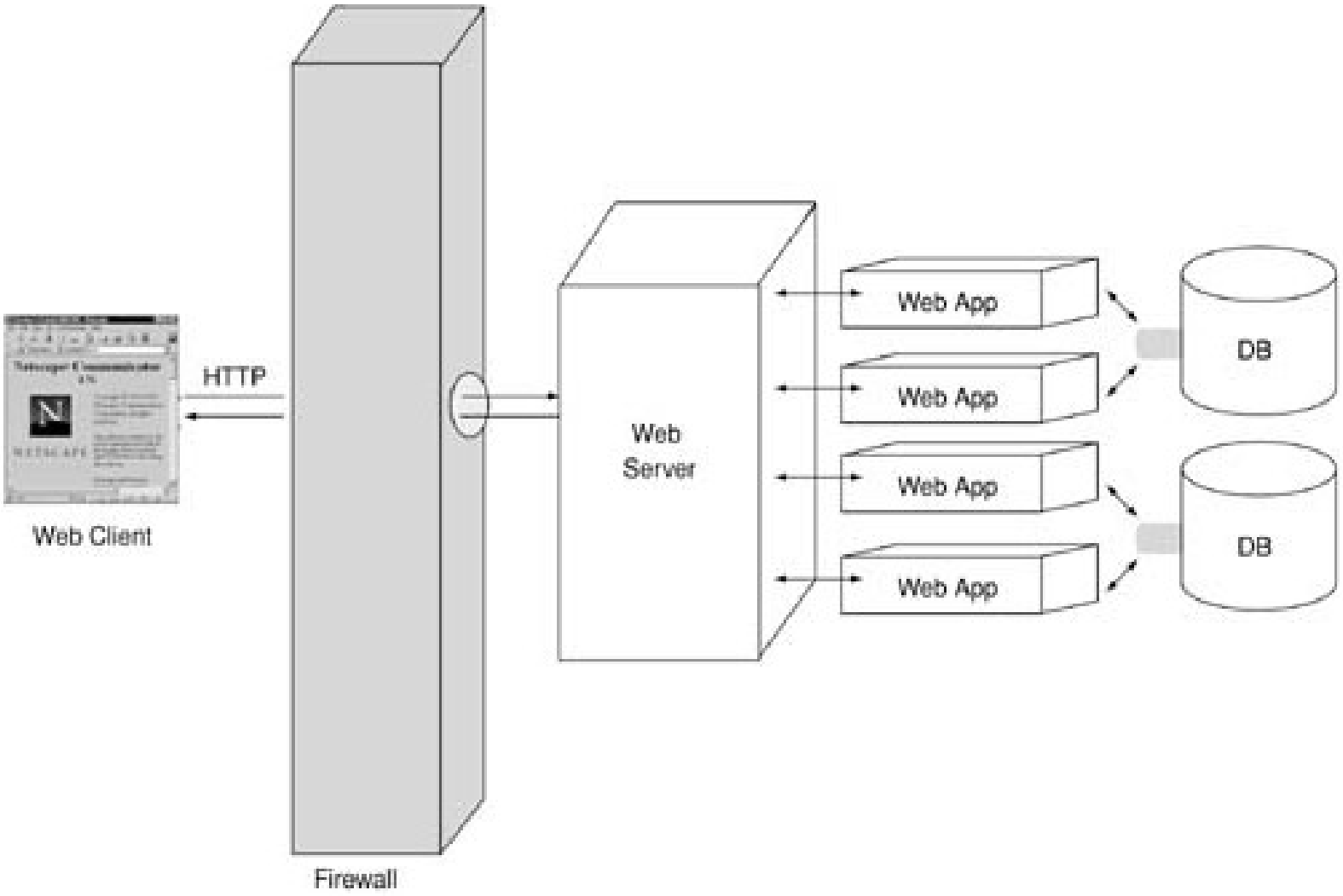
```
<form method=POST action="/admin/login.php">
<table border=0>
<tr>
<td>Username:</td> <td><input name=user type=text
  width=20></td>
</tr>
<tr>
<td>Password:</td> <td><input name=pass type=password
  width=20></td>
</tr>
</table>
<input type=submit value="login">
</form>
```

Komponen Aplikasi Web

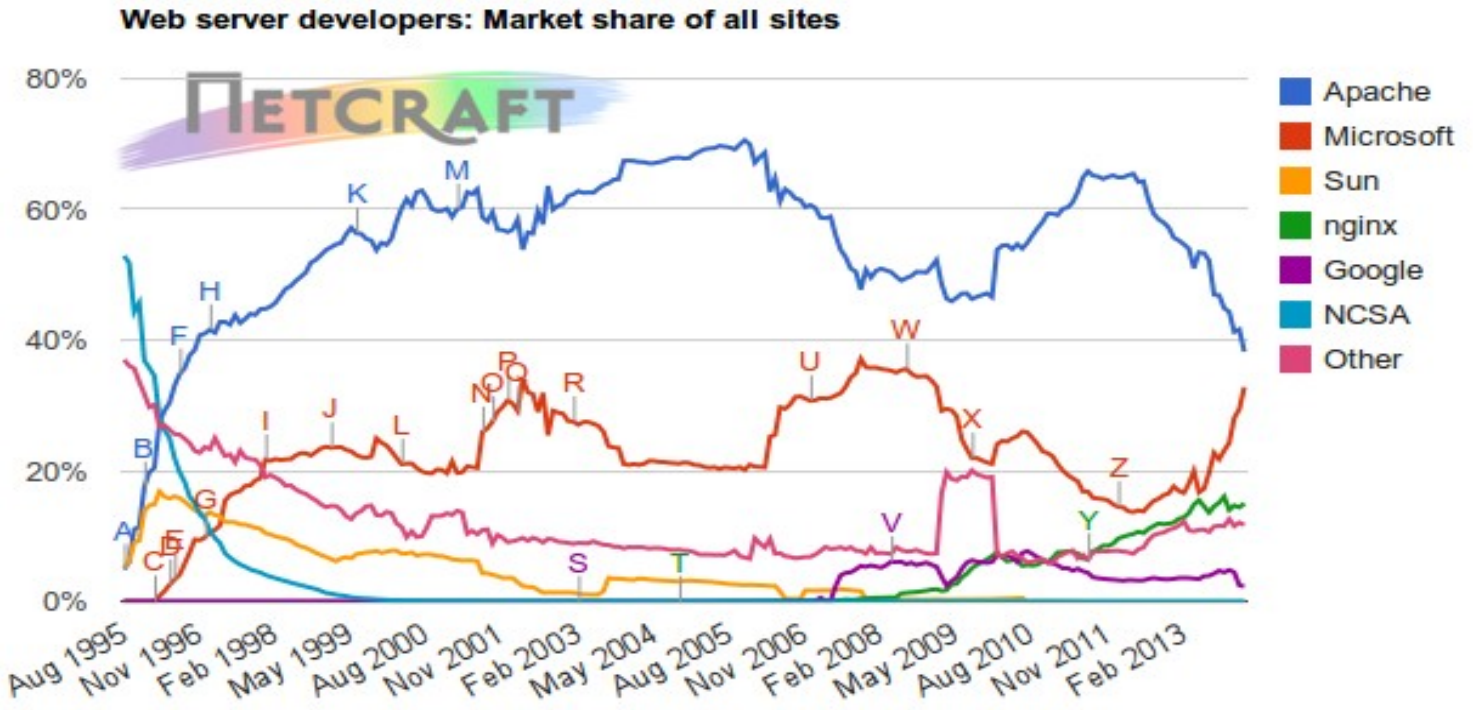
Suatu aplikasi web yang khas memiliki tiga komponen utama:

- Web server – presentation layer
- Web application – bussines logic
- Database server – core bussiness data

Komponen Aplikasi Web



Web server survey (2014)



Developer	January 2014	Percent	February 2014	Percent	Change
Apache	358,669,012	41.64%	351,700,572	38.22%	-3.41
Microsoft	253,438,493	29.42%	301,781,997	32.80%	3.38
nginx	124,052,996	14.40%	138,056,444	15.00%	0.60
Google	21,280,639	2.47%	21,129,509	2.30%	-0.17