Software Distribuït - T7 - WEB

Eloi Puertas i Prats

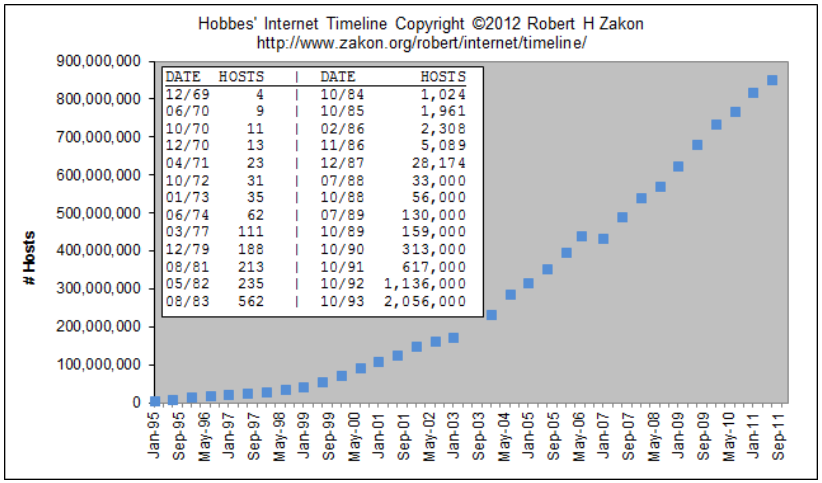
Universitat de Barcelona
Grau en Enginyeria Informàtica

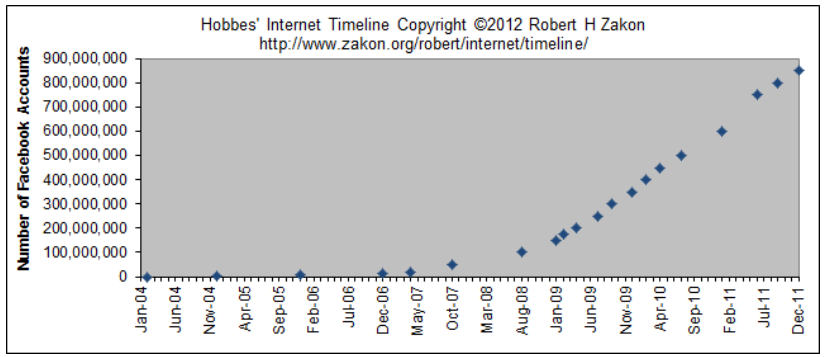
25 de març de 2020

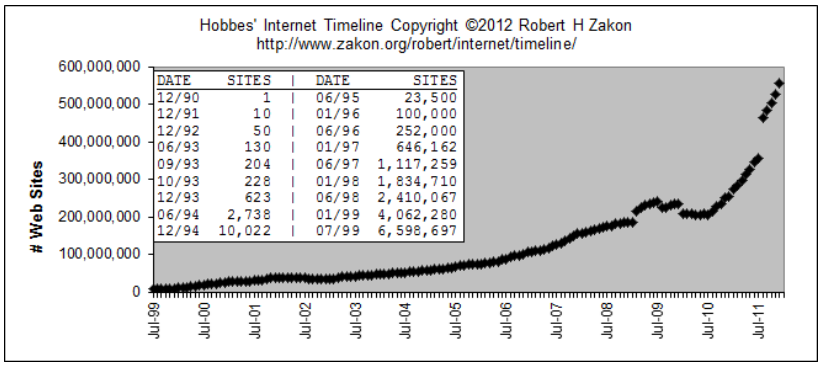
World Wide Web WWW

- De lluny, és el sistema distribuït més ben conegut.
- Formada per: servidors HTTP i clients, coneguts com web servers i web browsers.
- Aplicacions distribuïdes anteriors al WWW: mail, usenet, ftp, gopher ....
- WWW el crea Tim Berners-Lee l'any 1990 al CERN (Ginebra, Suïssa)

Evolució de la WWW







UNIVERSITAT DE BARCELONA

Funcionament bàsic de la WWW

- Bàsicament, la WWW és una aplicació client-servidor basada en el protocol HyperText Transfer Protocol (HTTP).
- Un servidor web és un servidor orientat a la connexió que implementa HTTP.
- Per defecte, el servidor corre sobre el port 80.
- Els usuaris fan servir un client WWW (navegador) en una màquina local.
- El client interactua amb el servidor web d'acord amb el protocol HTTP, sol·licitant un document.
- Si el document es troba en el servidor, el contingut del document és retornat al client, que el presenta a l'usuari.

UNIVERSITAT DE BARCELONA

Característiques de la WWW

Així doncs les tres peces clau que combina són:

- URL: Descriu la la localització dels documents.
- HTTP: Protocol que ens permet la sol·licitud i transmissió dels documents
- HTML: Defineix l'estructura d'un document d'hipertext.

URL

- Constitueix el sistema de noms i adreces de la web. Els Uniform Resource Identifiers (URI o URL) són strings curts que identifiquen recursos a la web: documents, imatges, fitxers descarregables, serveis i altres recursos.
- Permeten l'accés a recursos sota una varietat d'esquemes de noms i mètodes d'accés com HTTP, FTP, etc. uniformement.
- Esquema de un URI:
  `protocol://username:passwd@hostname:port/path/subdirs/resource?param1=value1&param2=value2`
- Exemples:
  - `http://www.ub.edu/dyn/cms/continguts_ca/menu_eines/noticies/index.html`
  - `http://translate.google.es/?q=hello+people`
  - `http://194.169.201.177:8085/live3.mp3`
  - `ftp://anonymous:anonymous@ftp.rediris.es/`
  - `mms://icatjazz.directe-wm.emissio.catradio.cat/reflector:34439?origin=extern`

HTML

- HTML (HyperText Markup Language) és un llenguatge de marcatge que serveix per crear documents per a que siguin obtinguts usant WWW.
- HTML es basa en SGML (Standard Generalized Markup Langugage) amb semàntiques apropiades per representar informació de molts diversos tipus, no només text.
- Llenguatge Formal definit per a la w3c (wwwconsortium). Molts navegadors però no són estrictes a l'hora de llegir HTML. Validador.

UNIVERSITAT DE BARCELONA

codi simple HTML

example.html

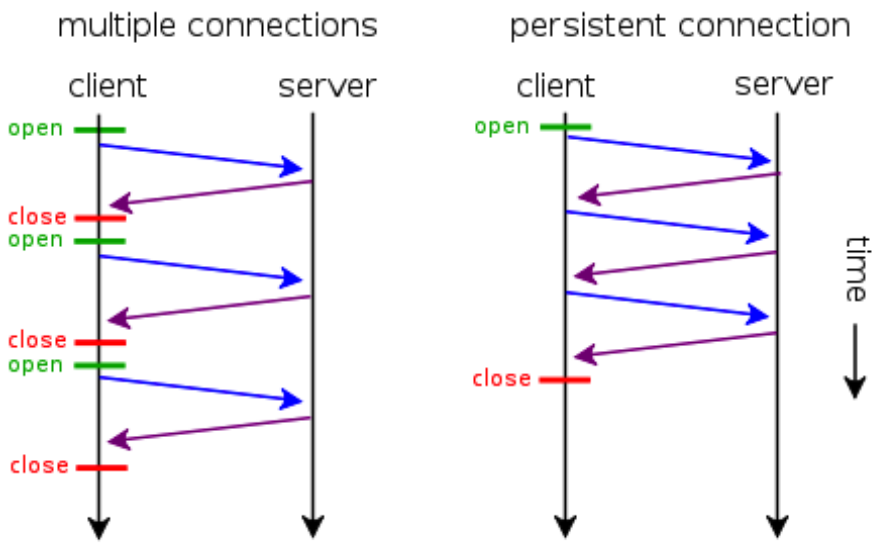
```
<HTML>
   <HEAD>
        <TITLE>A Sample Web Page</TITLE>
   </HEAD>
  <BODY>
   <center>
    <H1>My Home Page</H1>
    <b>Welcome to Eloi's page!</b>
    <br>
    <IMG SRC="./images/myPhoto.gif">
    <br>
    <! A list of hyperlinks follows.>
    <a href="./doc/publicacions.html"> My resume</a>.
    <br>
    <a href="http://www.ub.edu/">My university<a>
   </center>
  </BODY>
</HTML>
```

Protocol HTTP

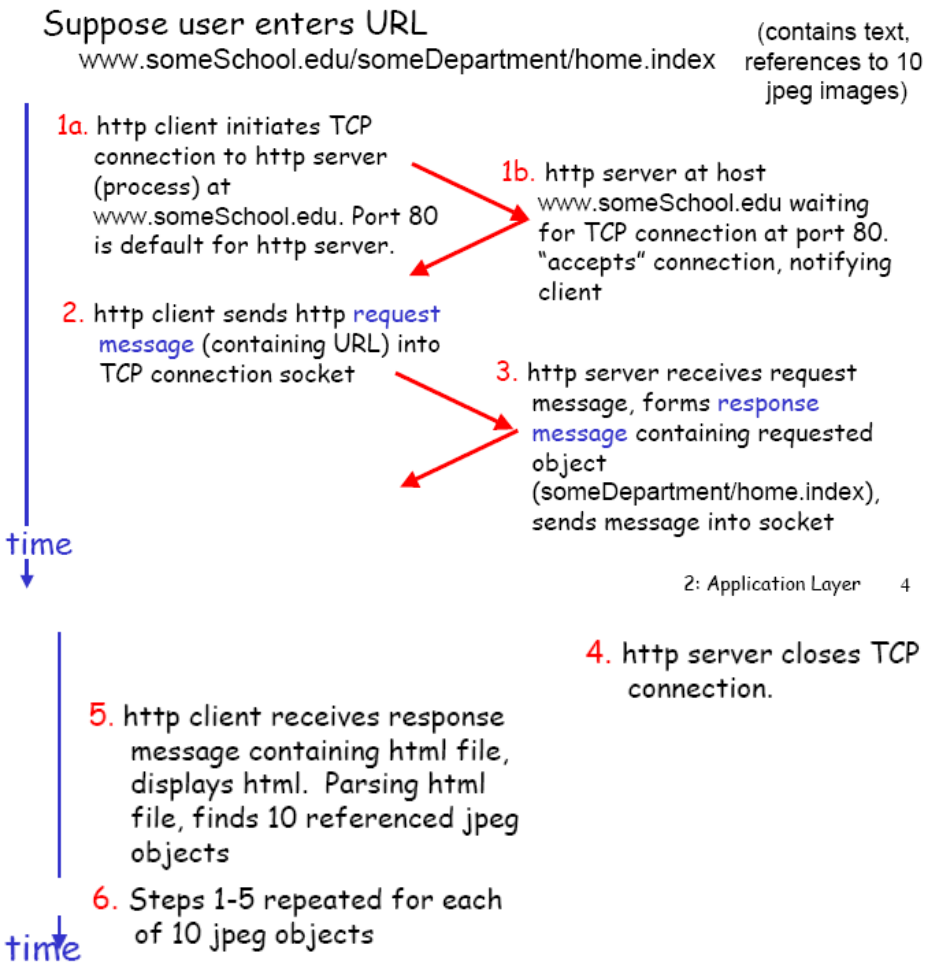
- Originàriament concebut per rebre i mostrar fitxers de texte. Ara s'ha extès per permetre la transferència de continguts de qualsevol tipus.
- La primera versió de HTTP (HTTP/0.9) era un protocol simple de transferència de dades sense format.
- El més utilitzat és la versió HTTP/1.0. Aquest s'ha anat extenent formant la versió coneguda com HTTP/1.1.

Protocol HTTP

- HTTP és un protocol orientat a la conexió, sense estat i d'anada i tornada (petició-resposta).
- A HTTP/1.0, cada connexió al servidor només permet una sola ronda de petició-reposta.
- A HTTP/1.1, en canvi permet persistència entre crides al mateix servidor

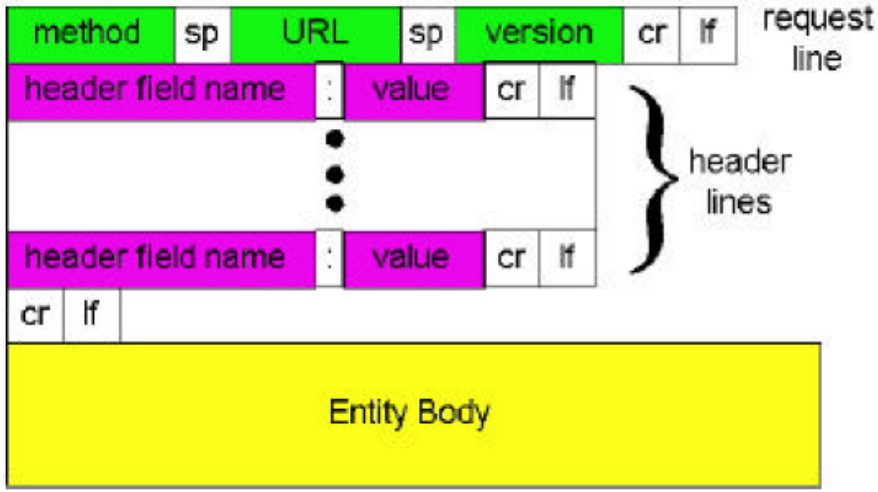


Protocol HTTP 1.0



Trames Protocol HTTP

- HTTP és un protocol basat en texte: la petició (**request**) i la resposta (**response**) són cadenes de caràcters.
- Cada request i response es composen d'aquestes parts, en ordre:
  1. Línia request/response.
  2. Secció capçalera.
  3. Línia en blanc.
  4. Cos.



Línia Request HTTP

S'envia un request un cop un client (navegador) estableix connexió amb el servidor web. La linia request és com segueix:

`<HTTP method><space><Request-URI><space><protocol specification>\r\n`

on:

- `<HTTP method>` és el nom d'un mètode especificat pel protocol.
- `<Request-URI>` és la URL d'un document web, o més general, un objecte a la web.
- `<protocol specification>` és l'especificació del protocol admesa pel client.
- `<space>` un caràcter espai en blanc.

Per exemple:

```
GET /index.html HTTP/1.0
```

Mètodes HTTP en el request del client

El mètode HTTP en una petició del client és una paraula reservada (en majúscules), que especifica quina operació del servidor, el client desitja fer. Pot ser segur(+) o no(-) i idempotent(+) o no(-).

+ + GET: per a recuperar el **contingut** de l'objecte web al que fa referència la URI especificada.
+ + HEAD: per a recuperar tan sols la **capçalera** d'un objecte web des del servidor, no el propi objecte.
- - POST: utilitzat per a **enviar dades** a un procés del servidor web.
- + PUT: s'utilitza per demanar al servidor **emmagatzemar** el contingut que s'adjunta amb la petició, a la ubicació del fitxer especificat per la URI en el servidor.

i també però menys utilitzats: DELETE, OPTIONS, TRACE, CONNECT

Capçalera del request del client

Afegeixen informació addicional d'interès per la intercomunicació entre navegador i servidor. Alguns dels keywords i values que poden aparèixer en la capçalera de la petició:

- Accept tipus: de contingut acceptables pel client
- User-Agent: especifica el tipus de navegador
- Connection: "Keep-Alive"si s'especifica, el servidor no tanca immediatament la connexió després d'enviar una resposta.
- Host: nom d'amfitrió del servidor
- Cookie, Authorization, Cache-Control, ...i moltes més ...

Cos del request del client

- Una petició, opcionalment, pot acabar amb un cos, que conté les dades que necessita transferir al servidor associades amb la sol·licitud.
- Per exemple, si el mètode especificat és **POST**, el cos conté les dades que es passen al procés de destí.

Exemple Request

web-sniffer: `http://www.maia.ub.es/%7Eeloi/example.html`

```
Connect to 161.116.83.1 on port 80 ... ok

GET ~eloi/example.html HTTP/1.1[CRLF]
Host: www.maia.ub.es[CRLF]
Connection: close[CRLF]
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.6; rv:11.0) Gecko/20100101 Firefox/11.0[CRLF]
Accept-Encoding: gzip[CRLF]
Accept-Charset: ISO-8859-1,UTF-8;q=0.7,*;q=0.7[CRLF]
Cache-Control: no-cache[CRLF]
Accept-Language: de,en;q=0.7,en-us;q=0.3[CRLF]
Referer: http://web-sniffer.net/[CRLF]
[CRLF]
```

web-sniffer plugin per firefox HTTP Fox

UNIVERSITAT DE BARCELONA

HTTP Response

En resposta a una sol·licitud rebuda d'un client, el servidor HTTP envia una resposta. Igual que el request, una resposta HTTP es compon de les següents parts:

1. La línia de resposta (o d'estat).
2. Secció capçalera
3. Un espai en blanc
4. El cos

Linia de Resposta del Response HTTP

té la següent forma:

`<protocol><sp><status-code><sp><description>\r\n`

Els codis d'estat són els següents:

- 100-199 Informatiu
- 200-299 Petició del client acomplerta amb èxit
- 300-399 Petició del client redirigida
- 400-499 Petició del client no acomplerta
- 500-599 Error en un procés del servidor.

Example 1:

```
HTTP/1.0 200 OK
```

Example 2:

```
HTTP/1.1 404 NOT FOUND
```

UNIVERSITAT DE BARCELONA

Linia de Capçalera del Response HTTP

Hi ha dos tipus de línies de capçalera de resposta:

- de resposta: Retorna informació sobre la resposta, el servidor o successius accessos al recurs sol·licitat
  - Refresh, Retry-After, Server, Set-Cookie ...
- de l'entitat: Retorna informació sobre els continguts del recurs sol·licitat pel client.
  - Content-Type, Content-Length, Content-Encoding, Content-Language ...

El cos de la respota HTTP

El cos de la resposta ve després de la capçalera i una línia en blanc, i conté el contingut de l'objecte Web sol·licitat, en el format especificat en el valor de la clau Content-Type de la capçalera.

```
<HTML>
   <HEAD>
        <TITLE>A Sample Web Page</TITLE>
   </HEAD>
  <BODY>
   <center>
    <H1>My Home Page</H1>
    <b>Welcome to Eloi's page!</b>
    <br>
    <IMG SRC="./images/myPhoto.gif">
    <br>
    <! A list of hyperlinks follows.>
    <a href="./doc/publicacions.html"> My resume</a>.
    <br>
    <a href="http://www.ub.edu/">My university<a>
   </center>
  </BODY>
</HTML>
```

Exemple Response

web-sniffer: http://www.maia.ub.es/%7Eeloi/example.html

```
HTTP/1.1 200 OK
Date: Wed, 25 Apr 2012 11:35:20 GMT
Server: Apache/2.2.16 (Debian)
Last-Modified: Thu, 12 Apr 2012 15:52:56 GMT
ETag: "2fe0005-175-4bd7d59714600"
Accept-Ranges: bytes
Content-Length: 373
Connection: close
Content-Type: text/html

<HTML>
   <HEAD>
<TITLE>A Sample Web Page</TITLE>
   </HEAD>
  <BODY>
   <center>
    <H1>My Home Page</H1>
    <b>Welcome to Eloi's page!</b>
    <br>
    <IMG SRC="./images/myPhoto.gif">
    <br>
    <! A list of hyperlinks follows.>
    <a href="./doc/publicacions.html"> My resume</a>.
    <br>
    <a href="http://www.ub.edu/">My university<a>
   </center>
  </BODY>
```

Client de protocol HTTP

JAVA

HTTPClient JAVA

Exemple d'ús:

```
java HTTPClient www.maia.ub.es 80 /%7Eeloi/example.html
```

URLBrowser JAVA

Exemple d'ús:

```
java URLBrowser www.maia.ub.es 80 /%7Eeloi/example.html
```

python

HTTPClient.py

UNIVERSITAT DE BARCELONA

Servidors web

October 2017

Product	Vendor	Percent
Apache	Apache	48.5%
nginx	NGINX, Inc.	35.4%
IIS	Microsoft	10.8%
LiteSpeed Web Server	LiteSpeed Technologies	2.9%
GWS	Google	1.1%

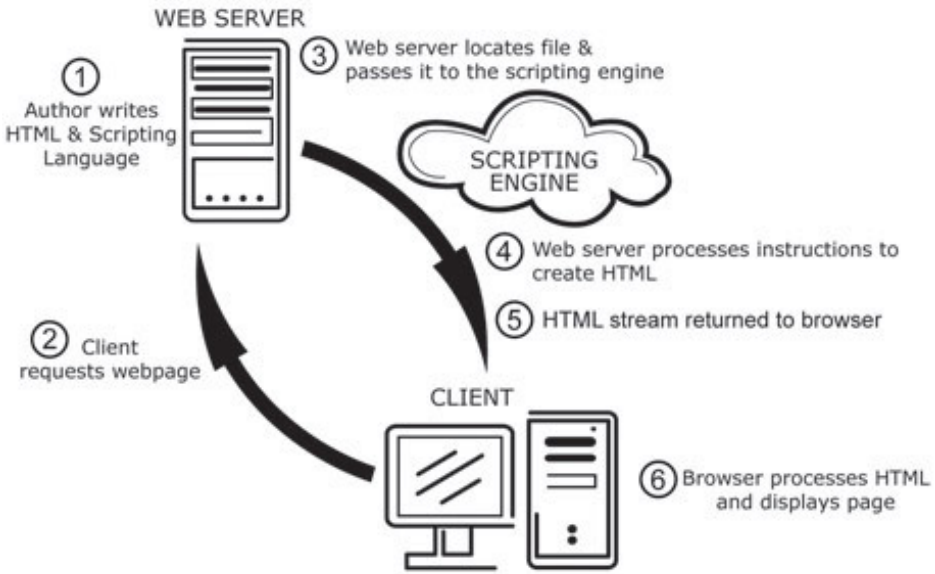
source : WIKIPEDIA

Aplicacions WEB

Al principi, l'HTTP va ser emprat per transferir continguts estàtics, com arxius de texte, arxius d'imatge... A mesura que la web va evolucionar, es va començar a utilitzar l'HTTP per a un fi no previst inicialment: fer que des del navegador un usuari pugui recuperar dades basant-se en informació dinàmica introduïda durant una sessió HTTP.
Un servidor HTTP genèric no té la lògica d'aplicació per tal d'anar a buscar les dades en el model de dades. Per tant es necessita d'un **procés extern** que tingui la lògica d'aplicació que faci d'intermediari.

- El **procés extern** s'executa a la màquina del servidor.
- accepta dades d'entrada des del servidor de web,
- executa la seva lògica d'aplicació per obtenir les dades
- retorna el resultat al servidor web, el qual el transmet el resultat al client.

Continguts web generats dinàmicament



Scripting Engine/	**Scripting Language**
CGI	C, Perl, ...
Zend Engine	PHP4
Tomcat	Servlets(Java/JSP)
WSGI	Python

Per eficiència, l'Scripting Engine s'executa com a una extensió del servidor web.

Executant un programa WEB

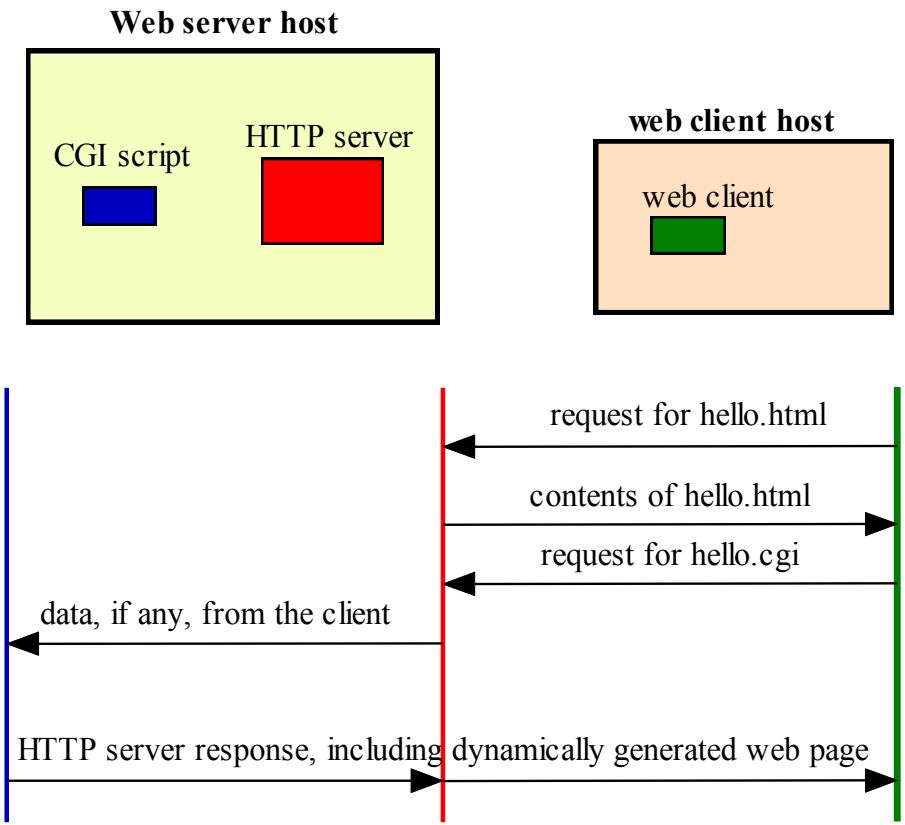
```php
<?php
echo "<head>\n";
echo "<title>Hello, World</title>\n";
echo "</head>\n";
echo "<body>\n";
echo "<font color=blue>\n";
echo "<h1>Hello, World</h1>\n";
echo "</font>\n";
echo "</body>\n";
?>
```

```perl
# Hello.pl
#   A simple Perl CGI script

print "Content-type: text/html\n\n";
print "<head>\n";
print "<title>Hello, World</title>\n
print "</head>\n";
print "<body>\n";
print "<font color=blue>\n";
print "<h1>Hello, World</h1>\n";
print "</font>\n";
print "</body>\n";
```

Executant un programa WEB

Una forma de cridar a un script web i enviar-li paràmetres d'entrades és mitjançant un formulari dins del codi d'una pàgina web.




```
<HTML>
<HEAD>
    <TITLE> Una pagina Web </title>
</HEAD>
<BODY>
<form ACTION="hello.php">
Feu clic al boto Enviar per a
activar l'script hello.php: <BR>
<input Type="submit" name="submit"
        VALUE="SUBMIT">
</FORM>
<a href = hello.pl> hello.pl </a>

</BODY>
</HTML>
```

Formularis web

El codi que genera un formulari web en HTML està encabit entre les etiquetes `<FORM>...</FORM>`
Dins l'etiqueta `<FORM>` es poden especificar els següents atributs:

- `ACTION=<` la URL absoluta o relativa que identifca l'objecte web al qual va destinat les dades del formulari quan s'enviï.`>`
- `METHOD=<` paraula reservada: POST o GET, la qual especifica la manera que l'objecte web espera rebre les dades enviades per l'usuari, anomenades *query data*.`>`

D'entre les etiquetes que apereixen dins del *FORM* ha d'haver-hi un botó tipus *submit*:
`<INPUT TYPE="Submit"NAME="submit"VALUE="SUBMIT≫`
Cadascun dels camps dins d'un formulari te un atribut *name* amb el tipus de camp que és i un *value* amb el valor que s'introdueixi en el camp.

Query Data

En el moment en que es prem el botó de tipus submit associat al formulari s'emet una petició HTTP a la URL que especifica l'ACTION i s'envia la *Query String*
La *Query String* és una seqüència de parells *name=value* separats pel caràcter **&**. On *name* és el nom de l'element d'entrada i *value* és el valor que l'usuari ha introduït. Els parells nom=valor són codificats utilitzant codificació URL, així alguns caràcters reservats són transformats a representació ASCII hexadecimal

Enviament de Query Data

La recollida de dades, incloent la codificació dels valors, es porta a terme pel navegador.
Quan el formulari és enviat per l'usuari, la query data es passa al servidor web segons el mètode que s'especifiqui en el formulari:

- GET: La URL especificada en el request s'exten amb la Query Data passada al final precedida del caràcter '?'

```
GET /cgi/getForm.cgi?name=John%20Cohen&color=red HTTP/1.1
```

!!! Com que la longitud de la línia request és limitada, aquest sistema no és adequat per enviar grans quantitats de dades.

- POST: Una petició HTTP POST ve seguida d'un cos de petició, el qual conté la Query Data per a enviar. La URL no es veu modificada

```
POST /cgi/postForm.cgi HTTP/1.0
Accept: */*
Connection: Keep-Alive
Host: myHost.someU.edu
User-Agent: Generic
```

Enviament de Query Data

La recollida de dades, incloent la codificació dels valors, es porta a terme pel navegador.
Quan el formulari és enviat per l'usuari, la query data es passa al servidor web segons el mètode que s'especifiqui en el formulari:

- GET: La URL especificada en el request s'exten amb la Query Data passada al final precedida del caràcter '?'

```
GET /cgi/getForm.cgi?name=John%20Cohen&color=red HTTP/1.1
```

- POST: Una petició HTTP POST ve seguida d'un cos de petició, el qual conté la Query Data per a enviar. La URL no es veu modificada

```
POST /cgi/postForm.cgi HTTP/1.0
Accept: */*
Connection: Keep-Alive
Host: myHost.someU.edu
User-Agent: Generic

name=John\%20Cohen\&color=red
```

Query Data

```
<HTML>
<HEAD>
    <TITLE> Query data </title>
</HEAD>
<BODY>
<FORM  action="getForm.pl" method="get">
What is thy NAME: <INPUT NAME="name"><P>
What is thy favorite color:
    <SELECT NAME="color">
        <option id="red"> RED </option>
        <option id="blue"> BLUE </option>
    </SELECT>
    <BR>
    <BR>
    <INPUT TYPE="Submit" NAME="submit" VALUE="SUBMIT">
</FORM>
</BODY>
</HTML>
```

Frameworks Aplicacions Web Generics

- Servlets+Beans+JSP
- **ASP.NET** ASP.Net MVC
- **JAVA** Apache Struts, Grails, Spring, JavaServer Faces...
- **PHP** Zend Framework, CakePHP.
- **Python** Django, Flask
- **Ruby** Ruby on Rails
- **Erlang** YAWS.
- **Node.js** Express.

Frameworks adicionals per crear Rich Internet Applications. Basats en AJAX

- GWT
- JavaFX
- DOJO

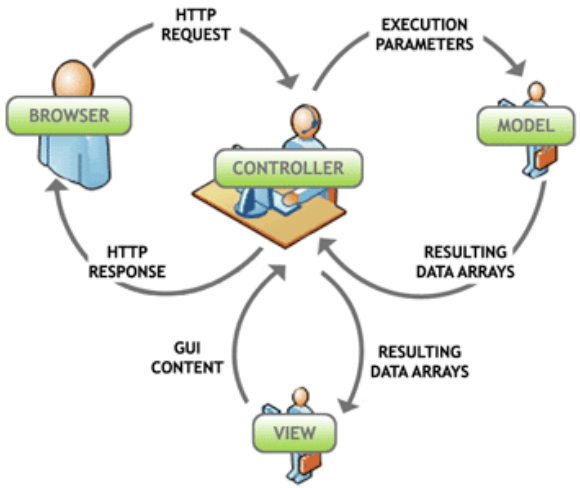
Frameworks especialitzats

- Content management systems (CMS): Alfresco, Umbraco, Drupal, Joomla!, Wordpress, Plone...
- e-commerce. Paquets Botigues online
- e-learning, Moodle.
- ERP online. (Enterprise resource planning)...

Frameworks Aplicacions Web

Basats en arquitectura Model View Controler.

- **Model** Model de dades que es manipulen segons les accions de l'usuari.
- **View** Diferents pàgines on es mostren els resultats i els elements que interectuen entre l'usuari i l'aplicació web.
- **Controller**
  - Un únic punt on es recullen totes les peticions.
  - Es gestiona la lògica de l'aplicació
  - S'envia el resultat a una vista, que s'encarregarà de mostrar-lo.



Flask (micro WSGI Framework)

hello.py

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello, World!'
```

```
$ export FLASK_APP=hello.py
$ flask run
* Running on http://127.0.0.1:5000/
```