The Web:
an architectural view

Needed Conceps:

TCP-IP protocol suite

Port

HTTP Overview

HTTP Requests
An HTTP request consists of
a request method, ("subprotocol" specification)
a request URL, (location)
header fields, (metadata)
a body. (data)

HTTP 1.1 defines the following request methods:
- GET: Retrieves the resource identified by the request URL
- HEAD: Returns the headers identified by the request URL
- POST: Sends data of unlimited length to the Web server
- PUT: Stores a resource under the request URL
- DELETE: Removes the resource identified by the request URL
- OPTIONS: Returns the HTTP methods the server supports
- TRACE: Returns the header fields sent with the TRACE request

HTTP 1.0 includes only the GET, HEAD, and POST methods. Although J2EE servers are required to support only HTTP 1.0, in practice many servers support HTTP 1.1.

HTTP Overview

HTTP Responses
An HTTP response contains a result code, header fields, and a body. The HTTP protocol expects the result code and all header fields to be returned before any body content.

Some commonly used status codes include:

- 100: Continue
- 200: OK

- 404: the requested resource is not available
- 401: the request requires HTTP authentication
- 500: an error occurred inside the HTTP server that prevented it from fulfilling the request
- 503: the HTTP server is temporarily overloaded and unable to handle the request

For detailed information on this protocol, see the Internet RFCs: HTTP/1.0 (RFC 1945), HTTP/1.1 (RFC 2616). (http://www.rfc-editor.org/rfc.html)

See also http://en.wikipedia.org/wiki/Http

HTTPS Overview

https is a URI scheme which is syntactically identical to the http: scheme normally used for accessing resources using HTTP. Using an https: URL indicates that HTTP is to be used, but with a different default port (443) and an additional encryption/authentication layer between HTTP and TCP.

This system was developed by Netscape Communications Corporation to provide authentication and encrypted communication and is widely used on the World Wide Web for security-sensitive communication, such as payment transactions.

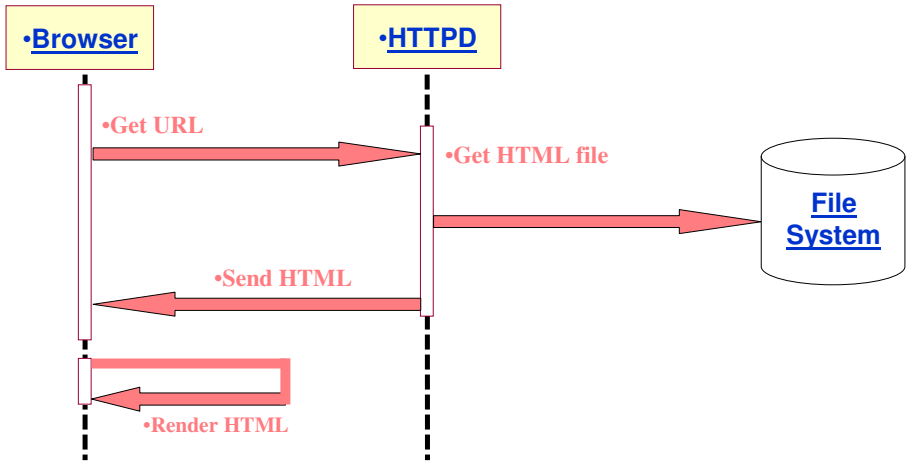
S-HTTP Overview

Secure hypertext transfer protocol' (S-HTTP) is an alternative mechanism to the https URI scheme for encrypting web communications carried over HTTP. S-HTTP is defined in RFC 2660.

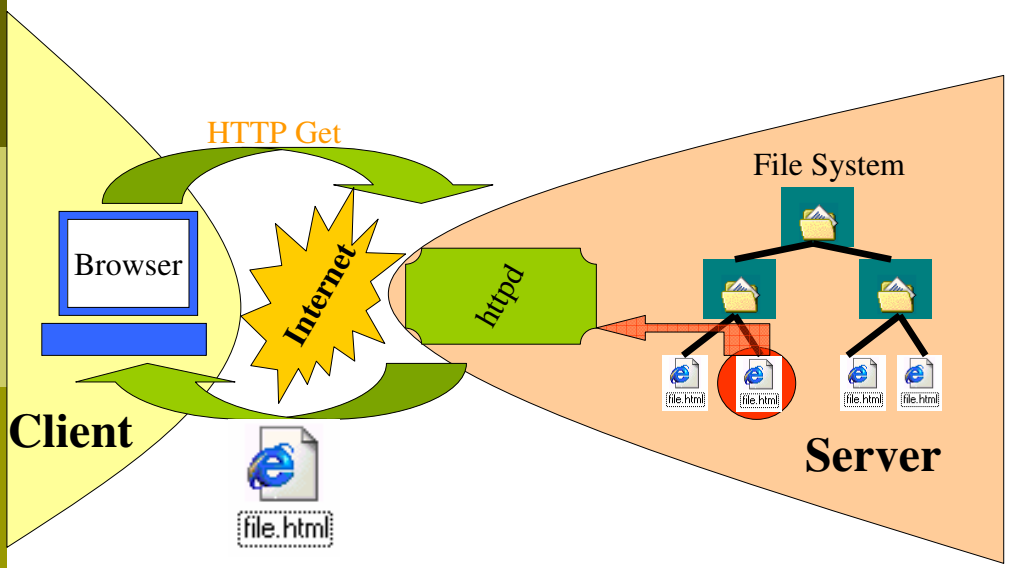
Web browsers typically use HTTP to communicate with web servers, sending and receiving information without encrypting it. For sensitive transactions, such as Internet e-commerce or online access to financial accounts, the browser and server must encrypt this information.

The https: URI scheme and S-HTTP were both defined in the mid 1990s to address this need. Netscape and Microsoft supported HTTPS rather than S-HTTP, leading to HTTPS becoming the de facto standard mechanism for securing web communications. S-HTTP is an alternative mechanism that is not widely used.

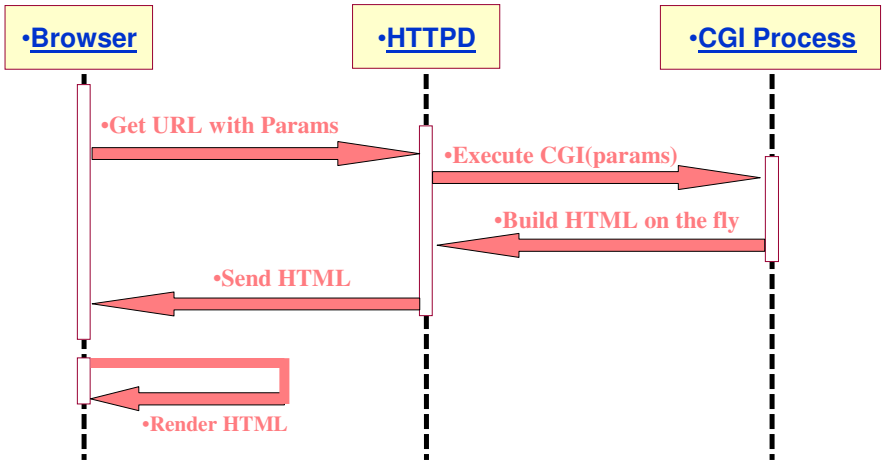
The primitive Web model



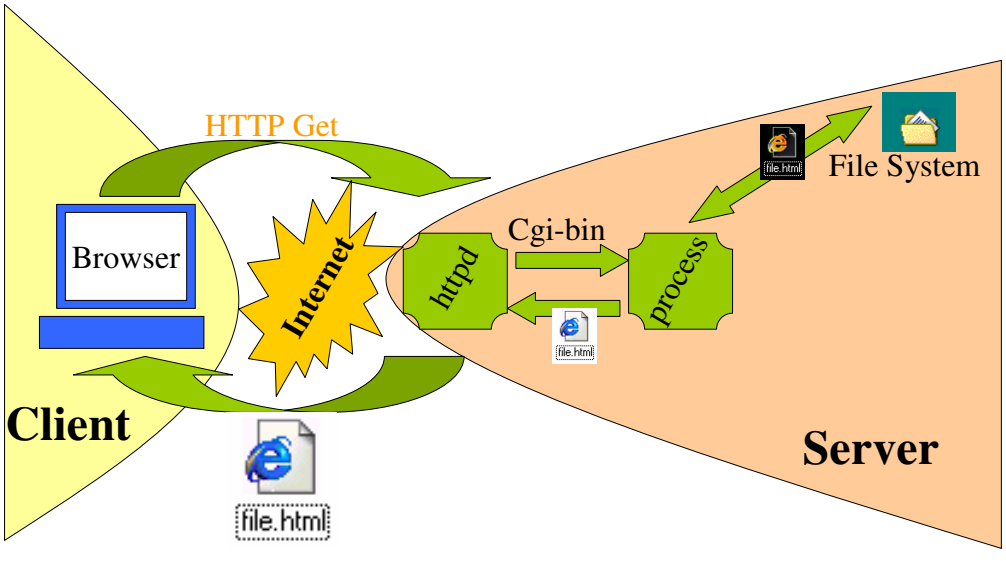
The primitive Web model



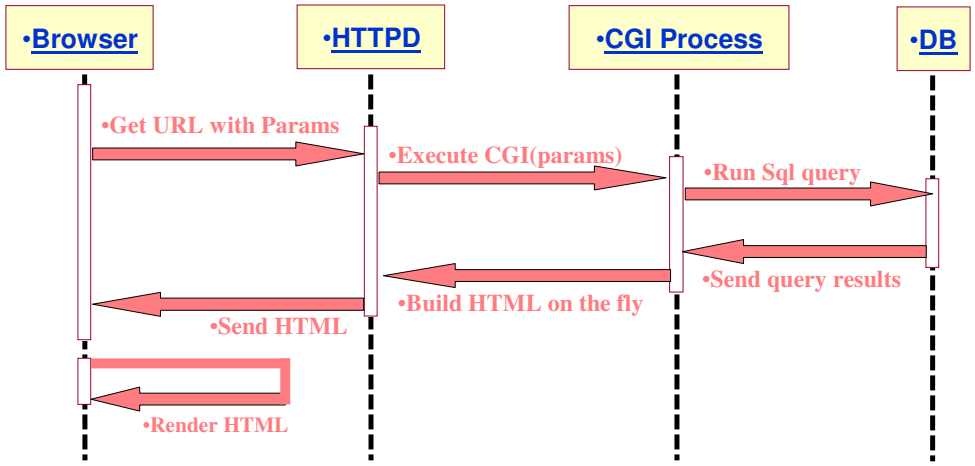
A simple interactive Web model



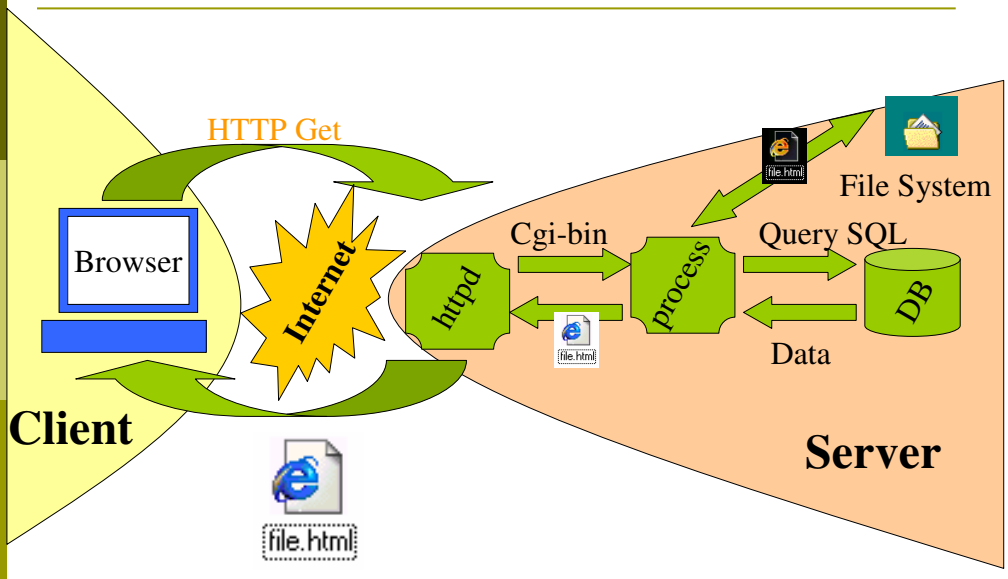
A simple interactive Web model

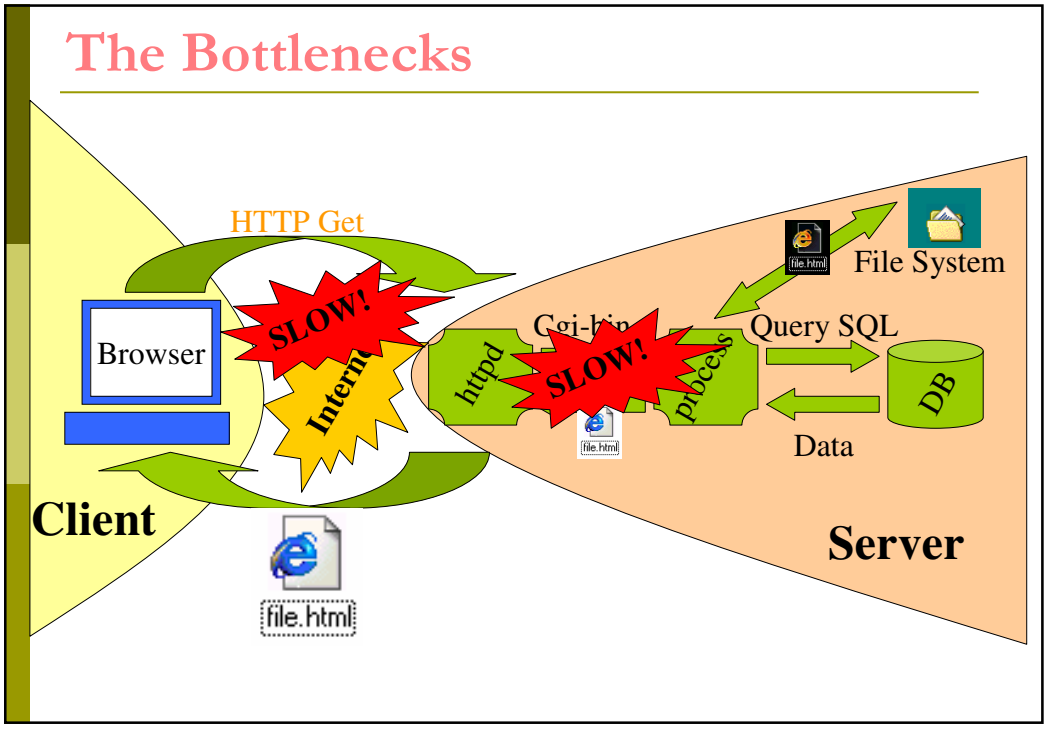


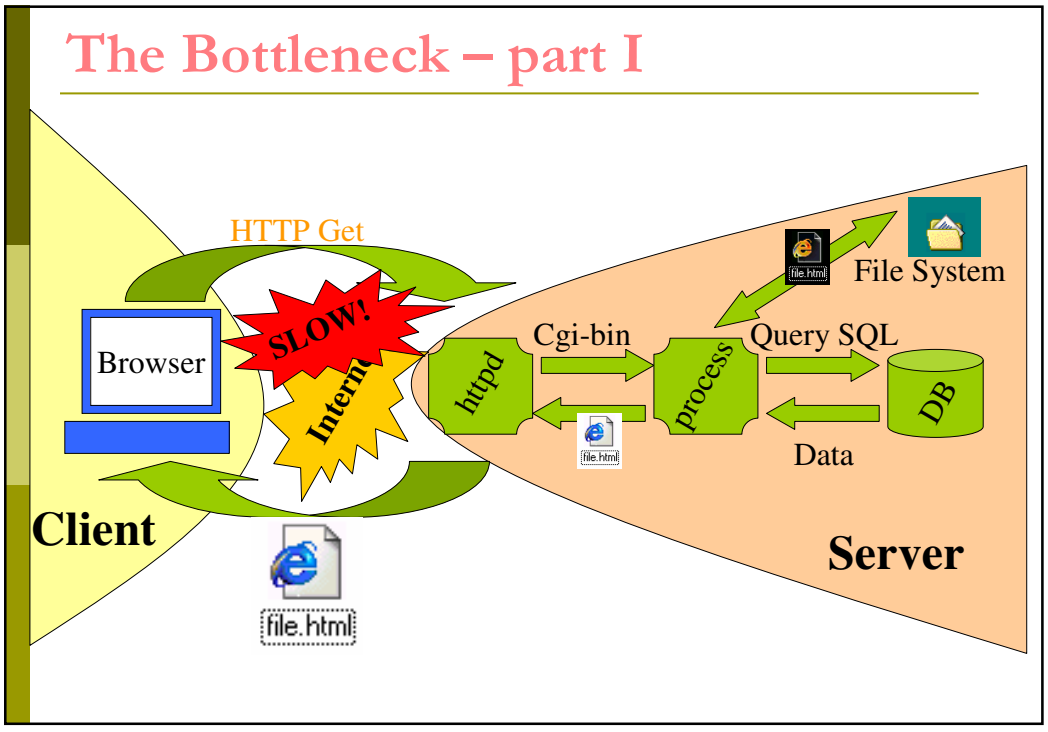
An evolved interactive Web model



An evolved interactive Web model

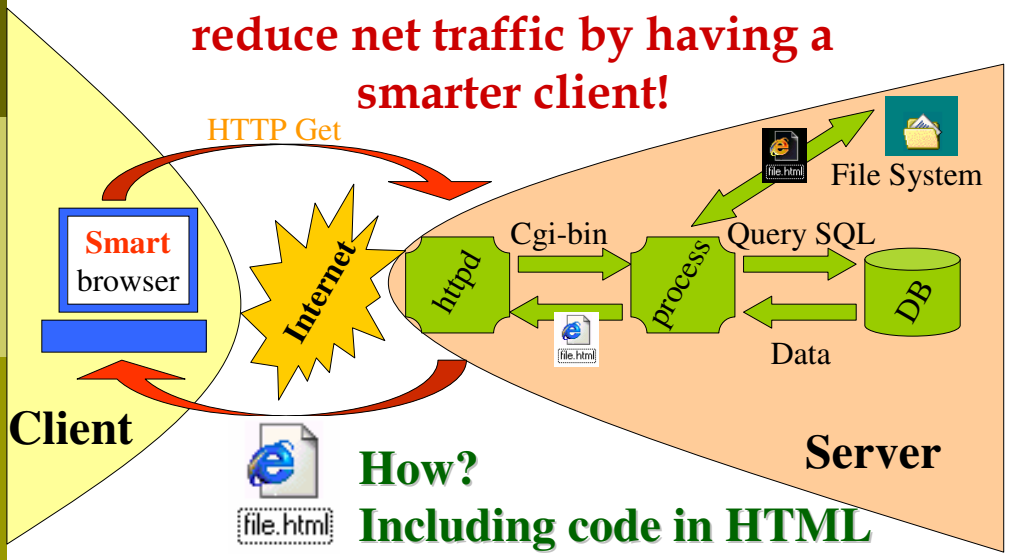



The Bottlenecks
HTTP Get
Browser
SLOW!
Internet
Client
file.html
Cgi-bin
httpd
SLOW!
process
Query SQL
DB
Data
File System
Server


The Bottleneck – part I
HTTP Get
Browser
SLOW!
Internet
Client
file.html
Cgi-bin
httpd
process
Query SQL
DB
Data
File System
Server

The solution:

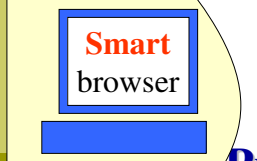
reduce net traffic by having a smarter client!



How?
Including code in HTML

Enabling technologies

(must be HW-OS-Browser independent!)



Client

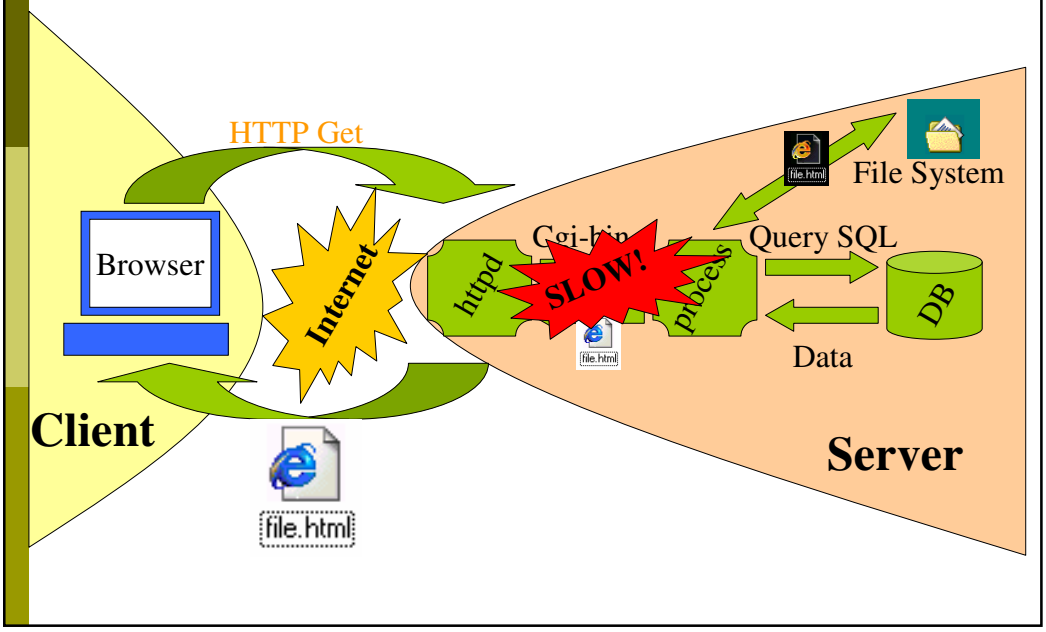
Scripting languages
- **Javascript**
- **Vbscript**
- **Perlscript**
- **Python**
- **…**

Interpreted:
Source code travels

Programming languages
- **Java**

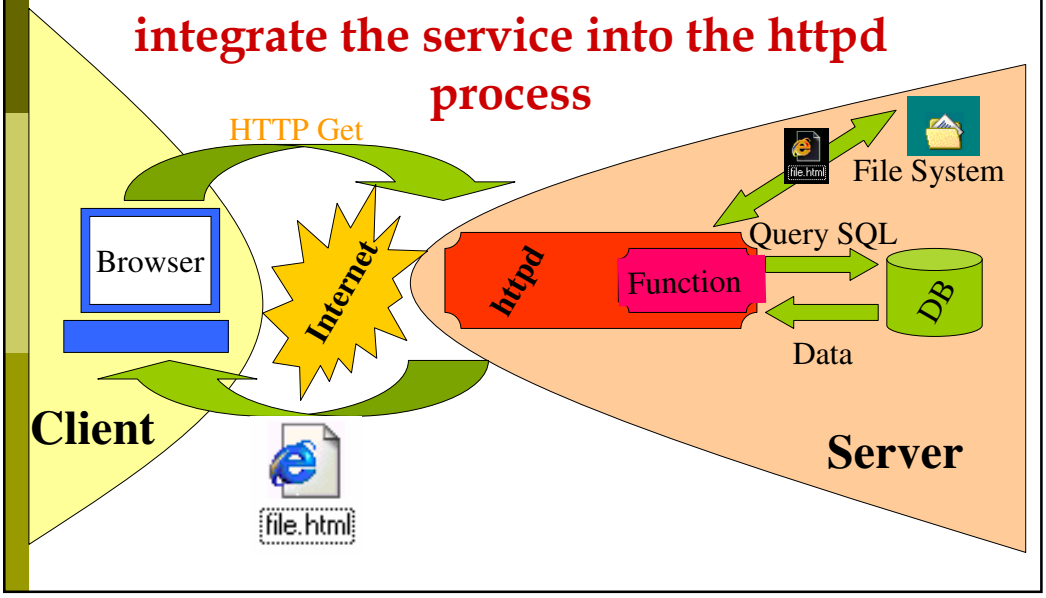
Compiled:
Executable travels

The Bottleneck – part II



The solution:

integrate the service into the httpd process



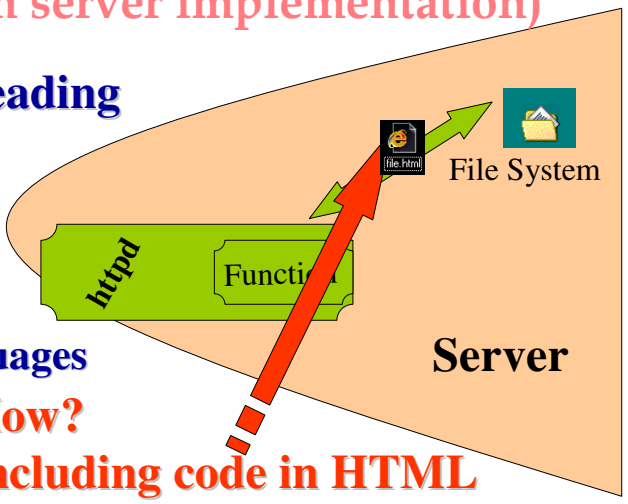
Enabling technologies

(depending on server implementation)

- **Multithreading**
- **DLL**
- **Servlets**

Using…

- **Scripting languages**
- **Programming languages**



How?
Including code in HTML

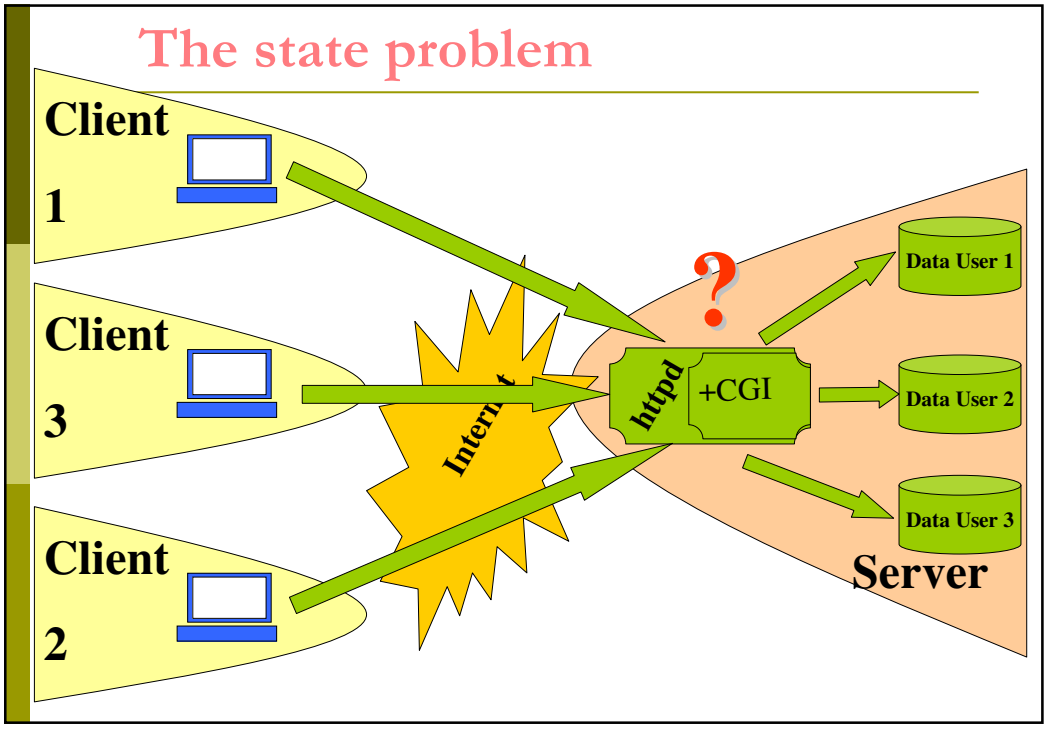
mixed client- and server-side scripting

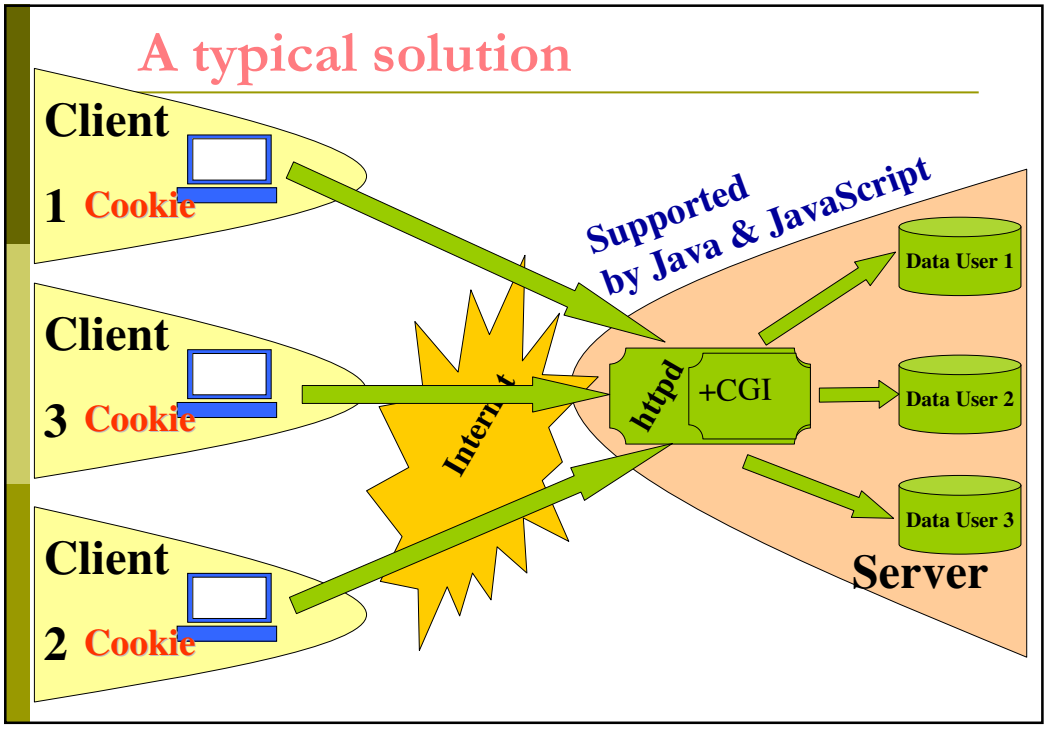
```
<HTML>
…
<SCRIPT LANGUAGE=VBScript RUNAT=SERVER>
…VBScript Commands…
</SCRIPT>
…
<% VBScript Commands %>
…
<SCRIPT LANGUAGE=JavaScript>
…JavaScript Commands…
</SCRIPT>
…
</HTML>
```

ASP Syntax

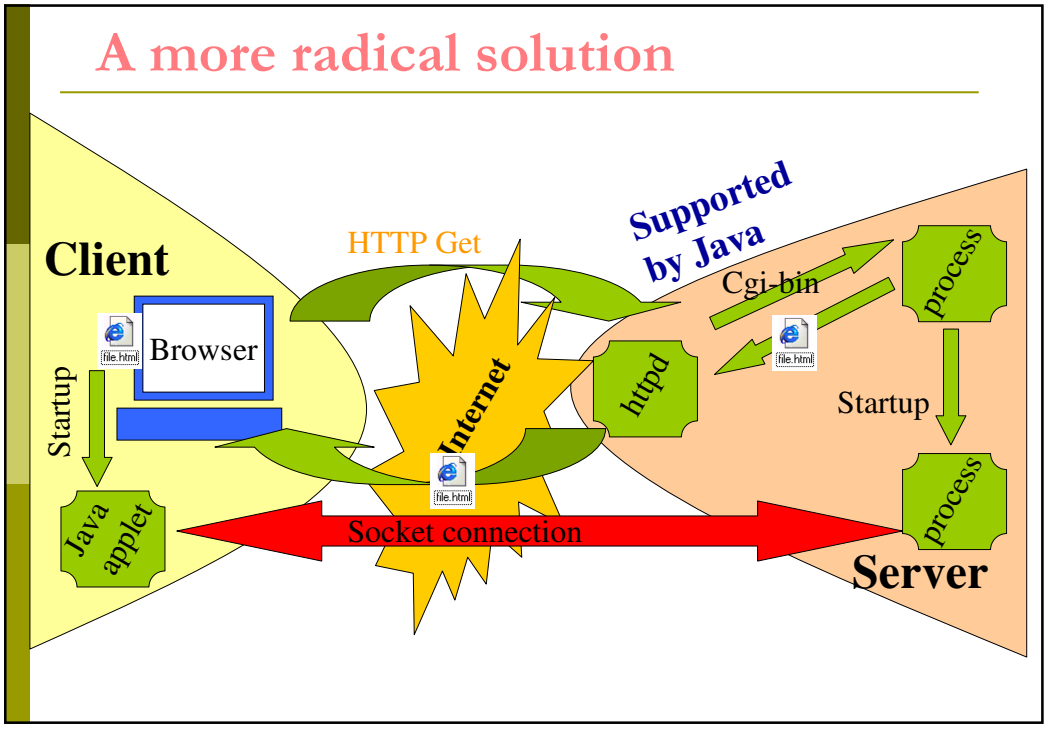
Code executed by the Server BEFORE the page is transferred over the Net

Code transferred to the client and interpreted by the Browser

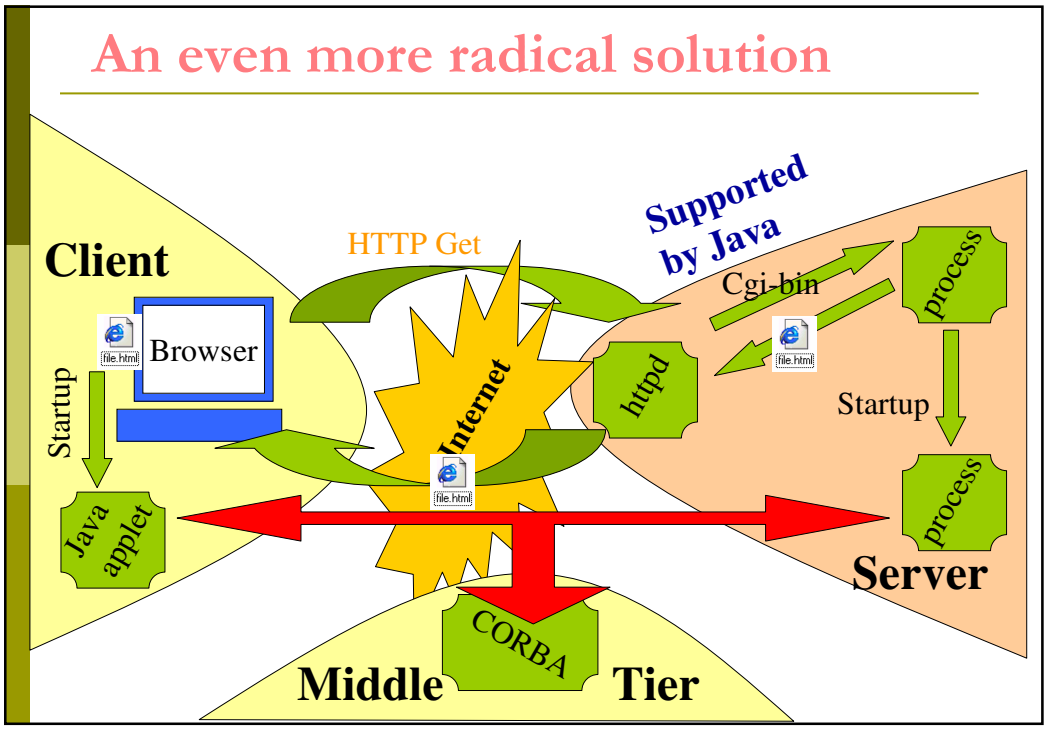

The state problem
Client 1
Client 3
Client 2
?
httpd
+CGI
Data User 1
Data User 2
Data User 3
Server


A typical solution
Client 1 Cookie
Client 3 Cookie
Client 2 Cookie
Supported by Java & JavaScript
httpd
+CGI
Data User 1
Data User 2
Data User 3
Server

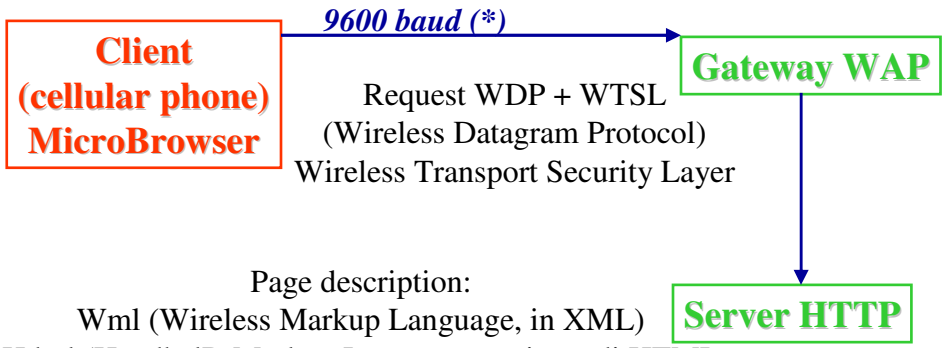
A more radical solution



An even more radical solution



WAP - Cenni

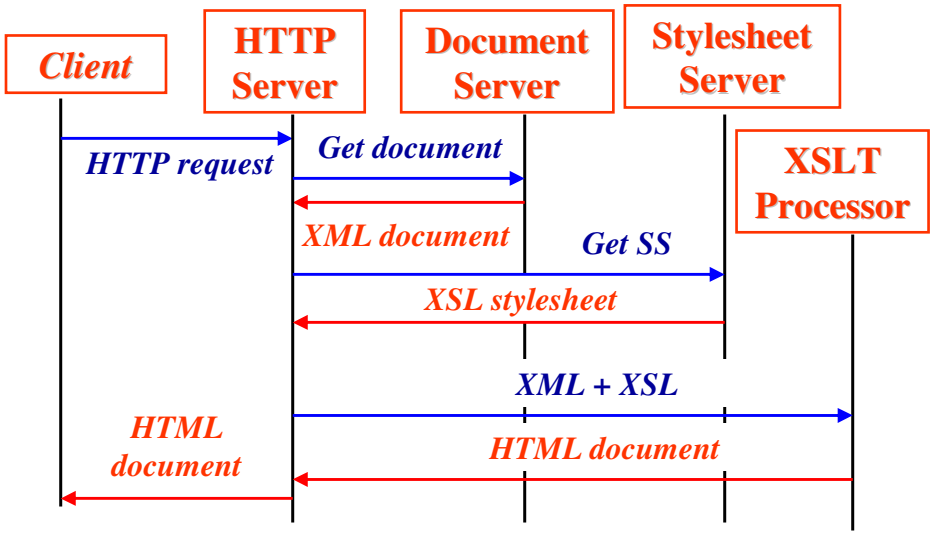


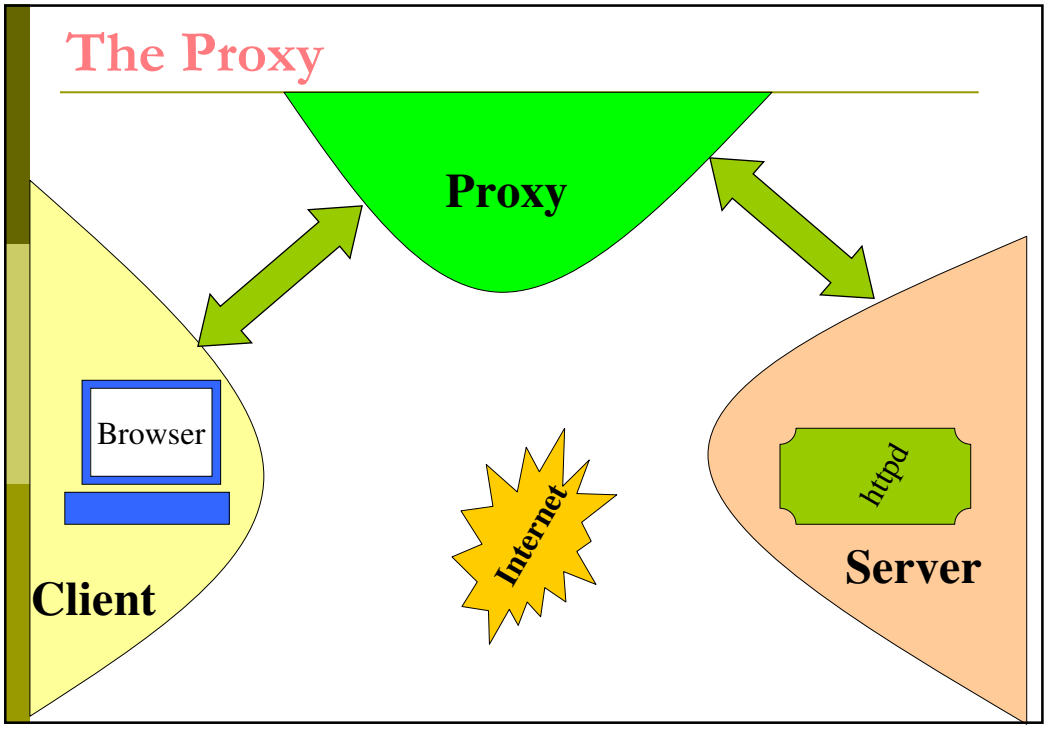
Page description:
Wml (Wireless Markup Language, in XML)
Hdml (HandhelD Markup Language, variante di HTML

More info: www.wapforum.org

(*) GPRS up to 56 Kbit/sec, UMTS up to 2 Mbit/sec
(Universal Mobile Telecommunication System)

XML Enabled HTTP Server




The Proxy
Proxy
Browser
Client
Internet
httpd
Server