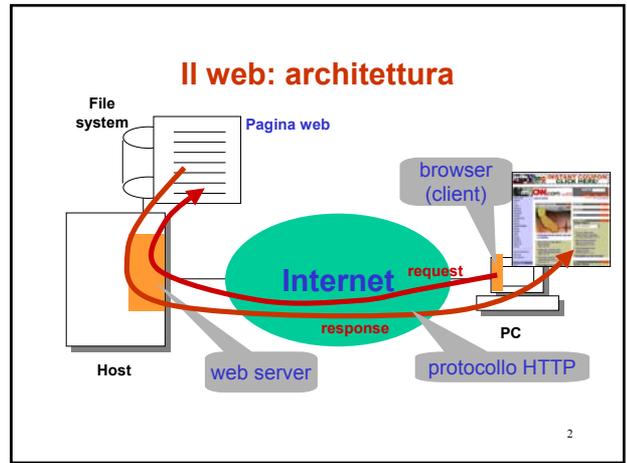


# ARCHITETTURA WEB



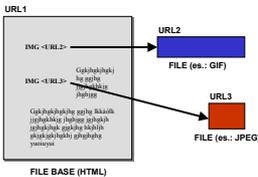
## Che cos'è una pagina web

Una pagina web è un insieme di **oggetti**, ciascuno dei quali è un **file** (es. file HTML, JPEG, GIF,...), indirizzabile attraverso il suo **URL**

Quello che vediamo con un browser:



Come è fatto realmente:



## HTML

HyperText Markup Language



```
<html>
<head>
<title>New Page 1</title>
</head>
<body>
<p>PAGINA DI ESEMPIO</p>
<p>
width="288" height="263"></p>
<p>&nbsp;</p>
</body>
</html>
```

## URL (Uniform Resource Locator)

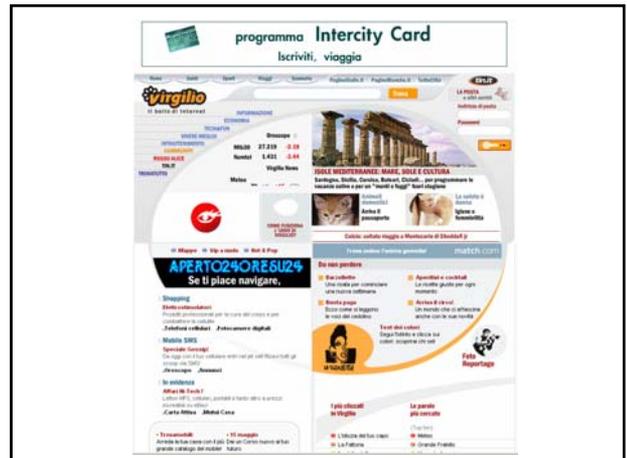
Ogni oggetto viene individuato attraverso un nome (URL) composto di tre parti:

`<protocollo>://<host name>/<pathname>`

Esempio:

`http://www.rpolillo.it/IUM/index.htm`

domain name



## Browser

- implementa il lato client del protocollo HTTP: dialoga con un web server per richiedere l'invio di una pagina web, poi
- la visualizza, interpretandone le istruzioni HTML, e
- gestisce l'interazione con l'utente
- Esempi:
  - Microsoft Internet Explorer
  - Netscape Communicator

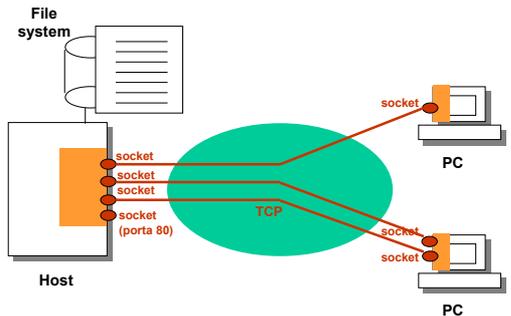
## Web server

- implementa il lato server del protocollo HTTP: memorizza gli oggetti web (identificabili col loro URL) e li invia a un browser che li richiede
- Esempi:
  - Apache
  - Microsoft Internet Information Server
  - Netscape Enterprise Server

## HTTP (HyperteXt Transfer Protocol)

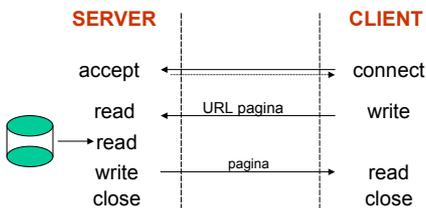
- Protocollo di comunicazione fra browser e web server
- Standardizzato dal W3C
- Due versioni:
  - HTTP/1.0 (fino al 1997)
  - HTTP/1.1 (dal 1998; compatibile con il precedente)
- Utilizza TCP come protocollo di trasporto

## Il web: architettura



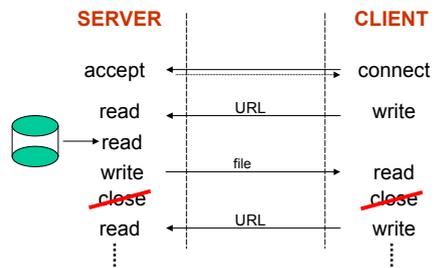
10

## Connessione non permanente (HTTP/1.0)



- Quindi per ogni connessione:
- viene inviato un singolo file
  - sono richiesti due RRT + il tempo di trasferimento file

## Connessione permanente (HTTP/1.1)



- Il server chiude la connessione quando non riceve richieste per un certo tempo (timeout)

## Messaggi di richiesta HTTP: formato

```
metodo pathname versionehtml <> request line
nomecampo:valorecampo<>
.....
nomecampo:valorecampo<>
<>
corpo
```



Legenda  
<>: return+line feed

## Esempio: GET

```
GET /IUM/index.html HTTP/1.1<>
Host: www.rpolillo.it<>
Connection: close<>
User-agent: Mozilla/4.0<>
Accept-language:fr<>
<>
```

*"Inviarmi il file /IUM/index.html dell'host www.rpolillo.it e poi chiudi la connessione. Sono un browser Mozilla 4.0, compatibile con HTTP 1.1. La lingua scelta dall'utente è il francese"*

## Esempio: POST

```
POST ...
.....
<>
dati
```

Allega alla richiesta di una pagina i dati forniti dall'utente in una form

## Esempio: HEAD

HEAD ...

Come GET, ma la pagina non viene inviata dal server (serve per test)

## Messaggi di risposta HTTP: formato

```
versione codicedistato frase <> status line
nomecampo:valorecampo <>
.....
nomecampo:valorecampo <>
<>
corpo
```



## Messaggi di risposta HTTP: esempio

```
HTTP/1.1 200 OK ← status line
Connection: close
Date: Thu, 06 Aug 2004 12:00:15 GMY
Server: Apache/1.3.0 (Unix)
Last-Modified: Mon, 22 Jun 2004 09:23:24 GMT
Content-Lenght: 7005
Content-Type: text/html
<>
dati ← return+line feed
```



## Alcuni codici di stato

- 200 OK
- 301 Moved permanently  
(il nuovo URL è specificato nello header Location: )
- 400 Bad request
- 404 Not found
- 505 HTTP Version not supported

## COOKIES

Sono file che contengono un identificativo di sessione (e il nome del server), che il client conserva su richiesta del server, e rispedisce al server ad ogni nuova connessione

**Esempio:**

Messaggio di risposta: Set-cookie: 15643546

Messaggio di richiesta: Cookies: 15643546

## COOKIES: problemi

- Il meccanismo non funziona se l'utente cambia macchina
- Possono essere disabilitati dall'utente del browser

