PARTE 11
HTTP E WEB

Modulo 1:
Introduzione al Web

Quando e perché nasce

Tim Berners-Lee (1989, CERN di Ginevra)

- “The current incompatibilities of the platforms and tools make it impossible to access existing information through a common interface, leading to waste of time, ...”
- “A link is specified as an ASCII string from which the browser can deduce a suitable method of contacting an appropriate server. When a link is followed, the browser addresses the request for the node [document] to the server.”

Ingredienti del Web

- **Meccanismi di comunicazione e naming di Internet**
  - Protocollo TCP/IP
  - Sistema DNS
- **Sistema client-server**
- **“Solo” tre nuovi standard**
  - URL: Sistema di indirizzamento delle risorse
  - HTML: Linguaggio di markup ipertestuale
  - HTTP: Protocollo per le richieste risorse

Modulo 2:
Meccanismi di naming

Ingredienti del Web

- **Meccanismi di comunicazione e naming di Internet**
  - Protocollo TCP/IP
  - Sistema DNS
- **Sistema client-server**
- **“Solo” tre nuovi standard**
  - URL: Sistema di indirizzamento delle risorse
  - HTML: Linguaggio di markup ipertestuale
  - HTTP: Protocollo per le richieste risorse

Meccanismi di naming del Web

L’insieme di tutti i meccanismi standard di naming delle risorse Web è l’Uniform Resource Identifiers (URI) che si compone di:

- Uniform Resource Locator (URL): specifica la locazione fisica delle risorse e le modalità di accesso
- Uniform Resource Name (URN): proposta per la rappresentazione univoca dei nomi delle risorse in modo da garantire persistenza e disponibilità
- Uniform Resource Characteristics (URC): proposta per la descrizione delle risorse basata su coppie attributo-valore. Es.: autore, data, copyright

Uniform Resource Locator (URL)

Il sistema di indirizzamento delle risorse è basato su Uniform Resource Locator (URL), un meccanismo standard per fare riferimento a tutte le risorse presenti nel Web:

- pagine (testo, immagini, suoni, video, ...)
- risultati di esecuzioni
- programmi eseguibili

Campi dell'URL

schema :// host.domain / pathname

http :// www.dsi.unimo.it / docenti/esami.html

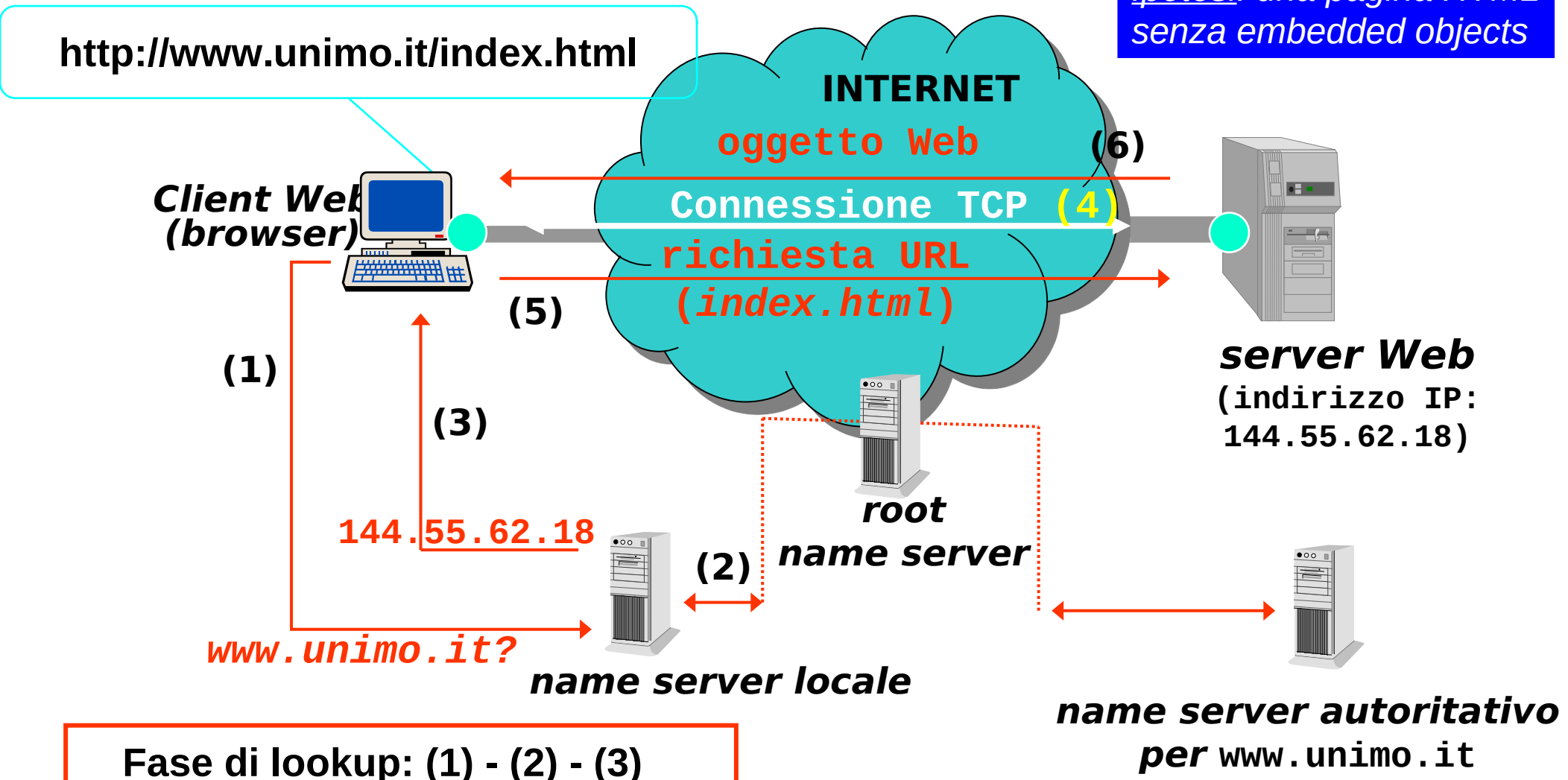
- **schema**: indica il modo con cui accedere alla risorsa, cioè quale protocollo bisogna usare per interagire con il server che controlla la risorsa. Il metodo di accesso più comune è ***HTTP*** (protocollo nativo del WWW per il recupero di risorse Web)
- **host.domain**: è l'hostname del nodo nel quale risiede la risorsa Web.
- **pathname**: identifica la risorsa presso il server Web.

Altre definizioni

- **Sessione utente: serie di richieste di risorse effettuate dallo stesso utente al medesimo sito Web**
- **Richiesta di risorsa (o pagina): una singola richiesta effettuata dall'utente (→ il tipico click), che tipicamente consiste di multiple richieste di hit inviate dal client dell'utente al sito Web**
- **Hit: una richiesta per un singolo oggetto effettuata dal client al server Web**

Richiesta di una risorsa Web
(cosa succede ad ogni click del mouse?)

Ipotesi: una pagina HTML senza embedded objects



Fase di lookup: (1) - (2) - (3)
Fase di connessione TCP: (4)
Fase di richiesta: (5) - (6)

Tipica risorsa Web

Pagina HTML + *embedded objects* (risorse di ogni tipo)

Modulo 3:
Linguaggio di markup

Ingredienti del Web

- **Meccanismi di comunicazione e naming di Internet**
  - Protocollo TCP/IP
  - Sistema DNS
- **Sistema client-server**
- **“Solo” tre nuovi standard**
  - URL: Sistema di indirizzamento delle risorse
  - HTML: Linguaggio di markup ipertestuale
  - HTTP: Protocollo per le richieste risorse

Documento Web

- **Una pagina è costituita da vari oggetti (testo, immagini binarie, …), detti *embedded objects*.**
- **Ad ogni oggetto corrisponde un file.**
- **Caratteristiche del testo tipico:**

  Rappresentazione in standard ASCII

  Si specifica sia il contenuto sia la rappresentazione (layout)

  Scritto nel **linguaggio di markup HTML**

Concetto di Ipermedia

- **I documenti Web tipicamente contengono un insieme di**

  testo

  immagini

  puntatori selezionabili ad altre risorse

  (audio)

  (video)

  Ipermedia

- **Uso mediante “point-and-click”**

Linguaggio di markup

Per permettere la corretta visualizzazione dell'informazione su qualsiasi piattaforma hardware connessa in rete è stato necessario definire un nuovo linguaggio di markup per la formattazione delle pagine:

- Fornisce delle linee guida generali per la rappresentazione del contenuto.
- Non specifica esattamente il formato e la posizione del testo, lasciando ai browser la definizione dei dettagli.

Due browser potrebbero visualizzare lo stesso documento in modo differente

Linguaggio HTML

- Sebbene siano stati proposti modifiche ed altri standard, a tutt'oggi *HyperText Markup Language* (HTML) nelle sue varie incarnazioni rimane il "**linguaggio del Web**".
- La pagina HTML è un file di solo testo ASCII.
- Il testo è *free-format*.
- Contenuto del testo e specifiche di formato possono essere inseriti nello stesso file.

Statement HTML

- La descrizione dei contenuti dell'ipertesto viene effettuata inserendo all'interno del testo stesso alcune istruzioni dette *marcatori* o *markup* o *tag* che producono le visualizzazioni e le azioni specificate.
- Gli statement HTML sono racchiuse tra parentesi angolari, nella forma **<tag>**, e vengono terminate da un tag di chiusura nella forma **</tag>**.

Immagini in un documento HTML

- Ciascuna immagine è contenuta in un file differente dal testo.

- Uso di espliciti tag per le immagini:

```
<IMG SRC="Pathname_del_file" >
```

- Possibilità di allineare le immagini al testo:

```
<IMG SRC="Pathname_del_file" align=middle>
```

Schema di un documento HTML

```
<HTML>
      <HEAD>
            <TITLE>
                  Titolo del documento
            </TITLE>
      </HEAD>
      <BODY>
            Corpo del documento
      </BODY>
</HTML>
```

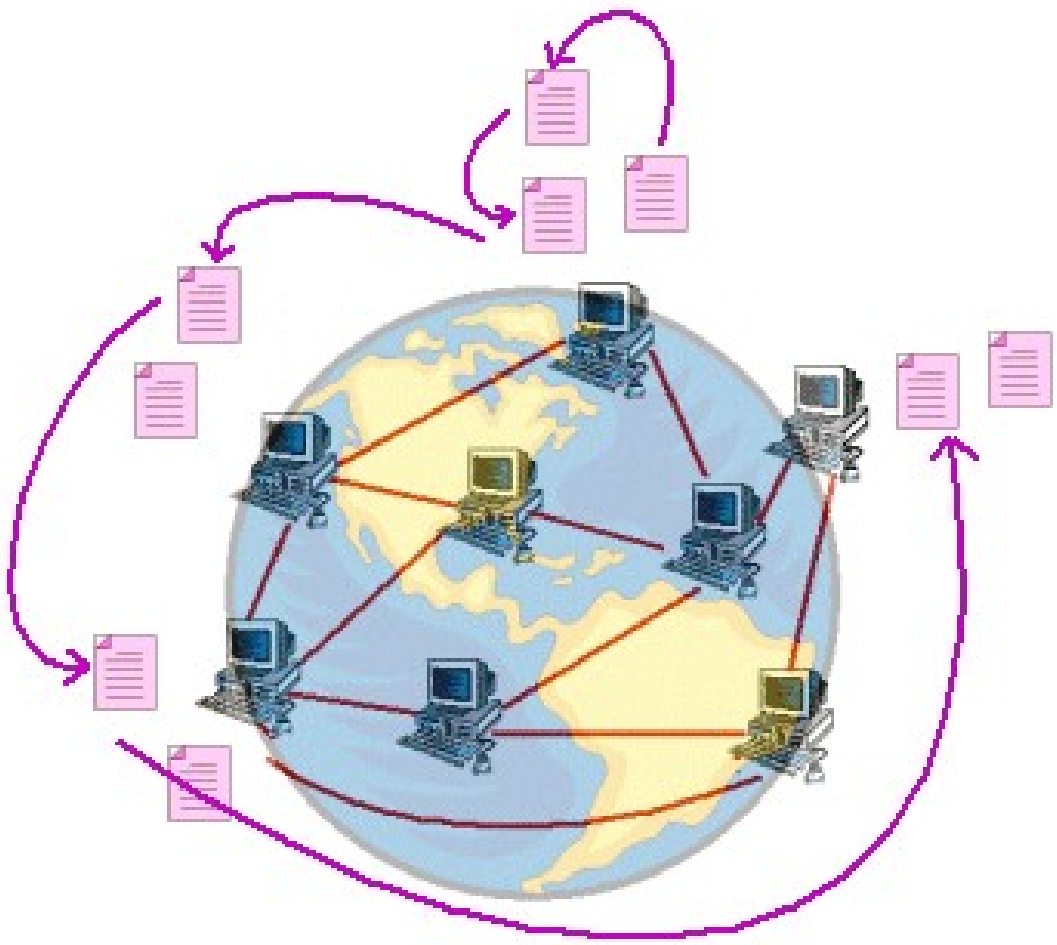
Tag àncora

- L’istruzione più innovativa dell’HTML è l’àncora delimitata dai tag <A>…</A>, in quanto tale elemento permette di trasformare un normale testo in ipertesto multimediale.
- Un àncora può far riferimento ad una sezione della stessa pagina oppure ad una qualsiasi risorsa (testuale, multimediale, eseguibile) presente sul Web, denotata mediante un URL che va inserito all’interno del tag àncora.

<A HREF="http://www.unimo.it/studenti/erasmus.html”>
Programma Erasmus </A>

- Il testo Programma Erasmus viene visualizzato in modo differente e risulta un link simbolico selezionabile via mouse.

Il tag àncora consente l'ipermedialità su scala geografica



Conseguenza dirompente dal punto di vista storico e sociale: l'informazione non è più organizzata in rigidi schemi (es., biblioteca), ma è accessibile (anche) mediante "link" decisi da singole persone, aziende, organizzazioni, …

Ogni link inserito tra una pagina Web e un'altra determina una nuova modalità di accesso all'informazione

Modulo 4
Cenni di HTTP

Ingredienti del Web

- **Meccanismi di comunicazione e naming di Internet**
  - Protocollo TCP/IP
  - Sistema DNS
- **Sistema client-server**
- **“Solo” tre nuovi standard**
  - URL: Sistema di indirizzamento delle risorse
  - HTML: Linguaggio di markup ipertestuale
  - HTTP: Protocollo per le richieste risorse

Protocollo HTTP

- **HyperText Trasmission Protocol (HTTP) è il protocollo che permette il reperimento delle risorse Web**
- **E’ un protocollo applicativo di tipo request/reply basato sulla suite di protocolli TCP/IP**
- **Tutti i client e server Web devono supportare il protocollo HTTP per poter scambiare richieste e risposte. Per questa ragione i client e i server Web sono chiamati anche client HTTP e server HTTP**

Protocollo HTTP

- **HTTP è un protocollo stateless (senza stato)**
  - il server non conserva nessuna informazione riguardante le richieste dei client passati
  - I protocolli che conservano lo stato sono complessi!
  - La storia passata (lo stato) deve essere memorizzata
  - Se il server/client subiscono un crash, la vista dello stato può essere inconsistente e deve essere ristabilita

Messaggi HTTP

Due tipi di messaggi:

- messaggi di **richiesta** HTTP
- messaggi di **risposta** HTTP

- **Messaggio di richiesta HTTP: ASCII**

Linea di richiesta (comandi GET, POST, HEAD)

Linee header

Carriage return, line feed indicanti la fine del messaggio

```
GET /somedir/page.html HTTP/1.1
Connection: close
User-agent: Mozilla/4.0
Accept: text/html, image/gif, image/jpeg
Accept-language:fr
```

(extra carriage return, line feed)

Richiesta HTTP

- **Una richiesta HTTP comprende**
  - metodo
  - URL
  - identificativo della versione del protocollo HTTP
  - insieme di extension header
- **Il metodo specifica il tipo di operazione che il client richiede al server. Il metodo più comune è GET che serve per acquisire pagine Web.**
- **Gli header contengono informazioni addizionali, quali la data e l'ora della comunicazione, il tipo di software utilizzato dal client, i tipi di dato che il browser è in grado di visualizzare, per un totale di circa 50 tipi di header differenti.**

Richiesta HTTP

- **Method: tipo di operazione richiesta dal client**
  - GET: richiesta di un oggetto
  - POST: il client richiede una pagina Web il cui contenuto è specificato dall'utente (es. richiesta ad un motore di ricerca) nel campo entity body
  - HEAD: il client richiede che il server invii soltanto l'header della risposta senza l'oggetto (usato per debugging dei Web server)
  - PUT, DELETE
  - LINK, UNLINK (HTTP 1.0)
  - TRACE, CONNECT, OPTIONS (HTTP 1.1)
- **URL: identificatore dell'oggetto richiesto**
- **version: versione del protocollo HTTP**

Metodi HTTP

Metodo	Richiesta	Versione
GET	Ricevere una risorsa dal server	>=0.9
HEAD	Ricevere il solo header di una risorsa	>=1.0
POST	Inviare dati al server	>=1.0
PUT	Inviare un oggetto al server	>=1.1
DELETE	Cancellare un oggetto dal server	>=1.1
LINK / UNLINK	Creare o eliminare collegamenti fra oggetti del server	<=1.0
TRACE	Individuare la catena dei server proxy	>=1.1
CONNECT	Creare connessione	>=1.1

Messaggio di richiesta HTTP (cont.)

Header lines

Connection: tipo di connessione richiesta dal client (persistente, non-persistente)

- User-agent: browser utilizzato dall'utente
- Accept: tipo di oggetti che il client accetta
- Accept-language: preferenza della lingua
- Accept-encoding, Accept-charset
- Host: specifica host che ha la risorsa (HTTP 1.1)
- Altri header per garantire la consistenza delle informazioni (es., If-Match o If-Modified-Since)

Risposta HTTP

- **Una risposta HTTP comprende, oltre al contenuto della risorsa richiesta, un header contenente l'identificativo della versione del protocollo HTTP, il codice di stato, l'informazione di stato in forma testuale, ed un insieme di possibili altre informazioni di risposta.**
- **Se la pagina richiesta, oltre al testo HTML, contiene altri oggetti, ciascuno di essi sarà identificato da un URL differente, per cui è necessario che il browser invii un esplicito messaggio di richiesta per ognuno degli elementi collegati alla pagina.**

Messaggio di risposta HTTP

Linea di stato (codice dello stato del protocollo, frase corrispondente)

Linee header

dati, ad es. il file HTML richiesto

```
HTTP/1.1 200 OK
Connection: close
Date: Thu, 06 Aug 1998 12:00:15 GMT
Server: Apache/1.3.0 (Unix)
Last-Modified: Mon, 22 Jun 1998
Content-Length: 6821
Content-Type: text/html

data data data data data ...
```

Messaggio di risposta HTTP (cont.)

Version: versione del protocollo HTTP

Status code, phrase: esito della richiesta (codice e frase)

- **1xx: Informazioni**
- **2xx: Successo**
- **3xx: Redirezione**
- **4xx: Errore del client**
- **5xx: Errore del server**

Risposte HTTP

Header line

- Connection: tipo di connessione usata dal server
- Date: data e ora della richiesta
- Server: tipo di Web server e di sistema operativo
- Last-Modified: data e ora creazione o modifica dell'oggetto (caching)
- Content-Length: dimensione in byte dell'oggetto
- Content-Type: tipo di oggetto (es. HTML, GIF, …)

- **Entity body: oggetto**

Gestione di richieste HTTP

- **Il Web è basato su un'architettura client-server**
- **Un processo particolare (Web server) si occupa di processare le richieste di risorse Web**
- **Le richieste sono veicolate attraverso il protocollo HTTP (Hyper Text Transfer Protocol)**
- **Ci sono modi diversi di implementare un Web server**
- **Ci sono diversi Web server**

API RESTful

- **Uso di HTTP per accedere a *risorse***
  - Supporto pr manipolazione di dati generici
  - Es. Tuple di un database
- **Risorse**
  - Mappate su URL
  - Es:
    /api/v1/movies/star-wars-a-new-hope
    /api/v1/directors/george-lucas
- **Valori tipicamente rappresentati con XML o JSON**

API RESTful

- **Valori tipicamente rappresentati con:**
  - Strutture XML

```
<director>
  <firstname>George</firstname>
  <lastname>Lucas</lastname>
</director>
```

  - Strutture JSON:

```
{firstname: ‘‘George’’, lastname ‘‘Lucas’’}
```

- **Manipolazione con metodi HTTP (CRUD)**
  - Create → metodo POST
  - Retrieve → metodo GET
  - Update → metodo PUT
  - Delete → metodo DELETE

Alcuni status code di risposta

HTTP Status Codes

For great REST services the correct usage of the correct HTTP status code in a response is vital.

1xx – Informational	2xx – Successful	3xx – Redirection	4xx – Client Error	5xx – Server Error
This class of status code indicates a provisional response, consisting only of the Status-Line and optional headers, and is terminated by an empty line	This class of status code indicates that the client's request was successfully received, understood, and accepted.	This class of status code indicates that further action needs to be taken by the user agent in order to fulfill the request.	The 4xx class of status code is intended for cases in which the client seems to have erred.	Response status codes beginning with the digit "5" indicate cases in which the server is aware that it has erred or is incapable of performing the request.
100 – Continue 101 – Switching Protocols 102 – Processing	200 – OK 201 – Created 202 – Accepted 203 – Non-Authoritative Information 204 – No Content 205 – Reset Content 206 – Partial Content 207 – Multi-Status	300 – Multiple Choices 301 – Moved Permanently 302 – Found 303 – See Other 304 – Not Modified 305 – Use Proxy 307 – Temporary Redirect	400 – Bad Request 401 – Unauthorised 402 – Payment Required 403 – Forbidden 404 – Not Found 405 – Method Not Allowed 406 – Not Acceptable 407 – Proxy Authentication Required 408 – Request Timeout 409 – Conflict 410 – Gone 411 – Length Required 412 – Precondition Failed 413 – Request Entity Too Large 414 – Request URI Too Long 415 – Unsupported Media Type 416 – Requested Range Not Satisfiable 417 – Expectation Failed 422 – Unprocessable Entity 423 – Locked 424 – Failed Dependency 425 – Unordered Collection 426 – Upgrade Required	500 – Internal Server Error 501 – Not Implemented 502 – Bad Gateway 503 – Service Unavailable 504 – Gateway Timeout 505 – HTTP Version Not Supported 506 – Variant Also Negotiates 507 – Insufficient Storage 510 – Not Extended

Examples of using HTTP Status Codes in REST

201 – When doing a POST to create a new resource it is best to return 201 and not 200.
204 – When deleting a resources it is best to return 204, which indicates it succeeded but there is no body to return.
301 – If a 301 is returned the client should update any cached URI's to point to the new URI.
302 – This is often used for temporary redirect's, however 303 and 307 are better choices.
409 – This provides a great way to deal with conflicts caused by multiple updates.
501 – This implies that the feature will be implemented in the future.

Special Cases

306 – This status code is no longer used. It used to be for switch proxy.
418 – This status code from RFC 2324. However RFC 2324 was submitted as an April Fools' Joke. The message is *I am a teapot.*

Key	Description
Black	HTTP version 1.0
Blue	HTTP version 1.1
Aqua	Extension RFC 2295
Green	Extension RFC 2518
Yellow	Extension RFC 2774
Orange	Extension RFC 2817
Purple	Extension RFC 3648
Red	Extension RFC 4918

BARONE, BUDGE & DOMINICK
Business • Technology • Solutions

REST is a style of development for exposing data in such a way that it is easy to consume, easy to produce and makes use of HTTP.
REST is not a replacement for SOAP or WS* but an alternative to solve certain problems better.

Architecture

Versioni protocollo HTTP

- **Versione 0.9**
  - Solo metodo GET
- **Versione 1.0**
  - Metodi GET, HEAD, POST
  - Metodi aggiuntivi: PUT, DELETE, LINK, UNLINK
  - Connessioni non persistenti
    (1 richiesta HTTP → 1 connessione TCP)
- **Versione 1.1**
  - Rimozione metodi LINK, UNLINK
  - Connessioni persistenti per default
    (Tante richieste HTTP in 1 connessione TCP)

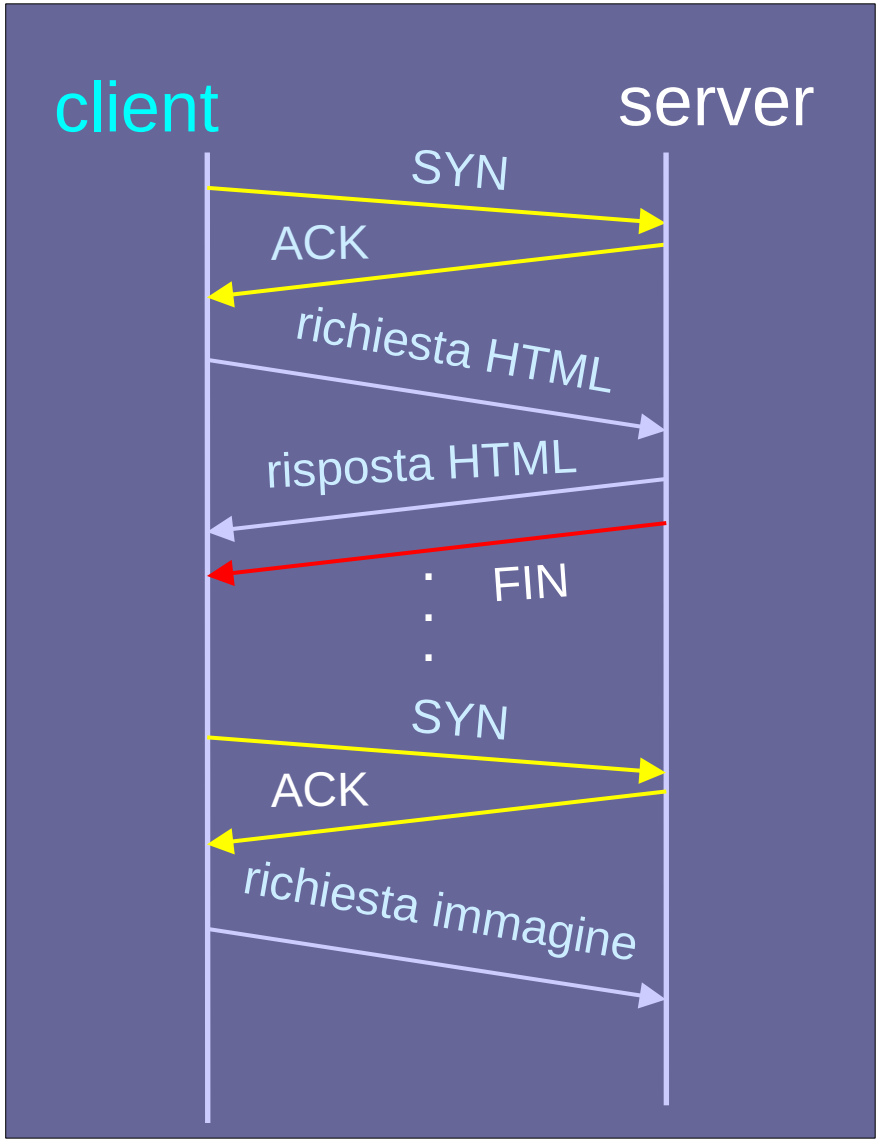
Connessioni TCP nell'HTTP 1.0

Connessione non-persistente: ogni oggetto viene trasferito usando una nuova connessione TCP

Versione HTTP/1.0 (RFC 1945)

- problema: three-way handshake del TCP: per ogni trasferimento di oggetto, in più anche un round-trip time per instaurare la connessione
- problema: slow-start del TCP (la finestra ha dimensione pari a 1 all'inizio di ogni nuova connessione)

Soluzione parziale: alcuni browser creano *simultaneamente* connessioni TCP multiple (una per oggetto), dopo aver analizzato i riferimenti presenti nel file HTML



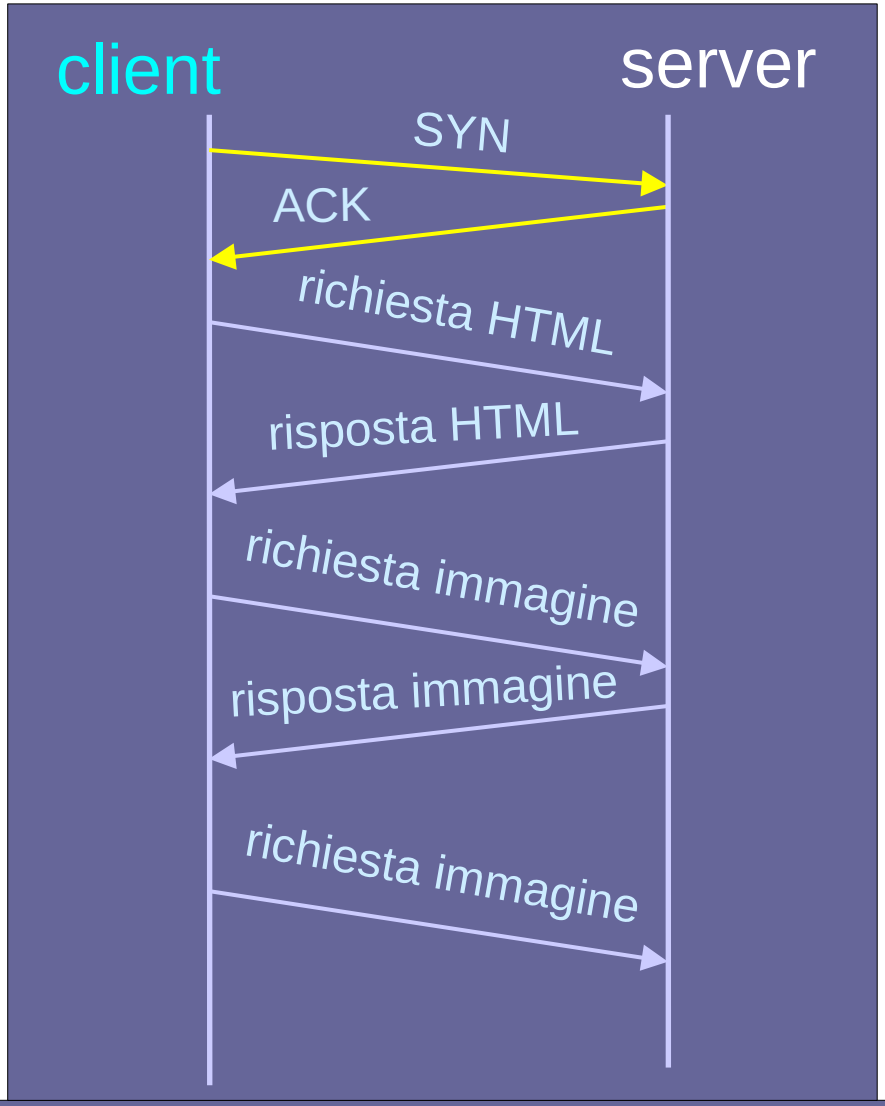
Connessioni TCP nell'HTTP 1.1

Connessione persistente: un numero multiplo di oggetti (ad es. tutti gli oggetti che compongono una pagina) vengono trasferiti entro una stessa connessione TCP

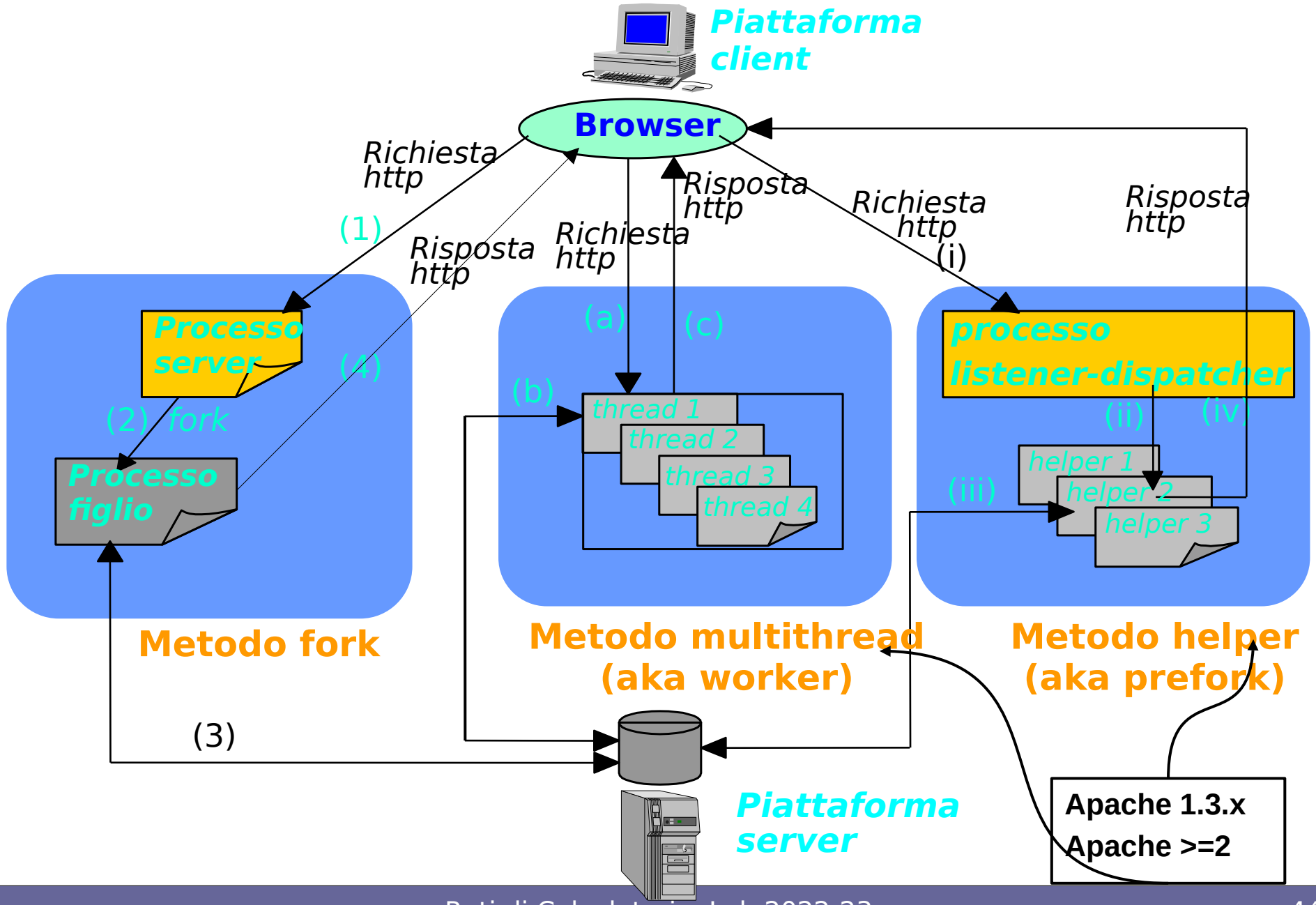
Versione HTTP/1.1 (RFC 2616)

- three-way handshake del TCP: solo per instaurare la connessione iniziale
- controllo di congestione a regime

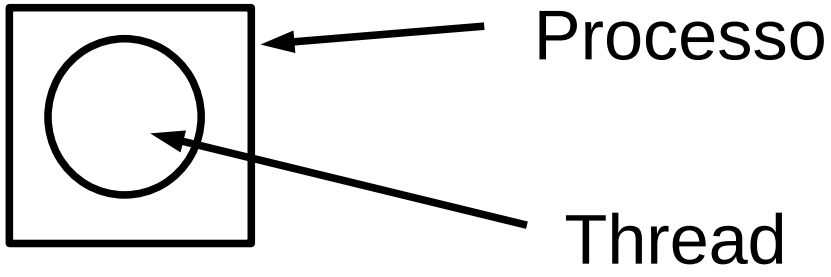
Pipelining in HTTP/1.1: il browser, dopo aver ricevuto ed analizzato il file HTML, invia più richieste consecutive sulla stessa connessione TCP senza aspettare di ricevere la risposta



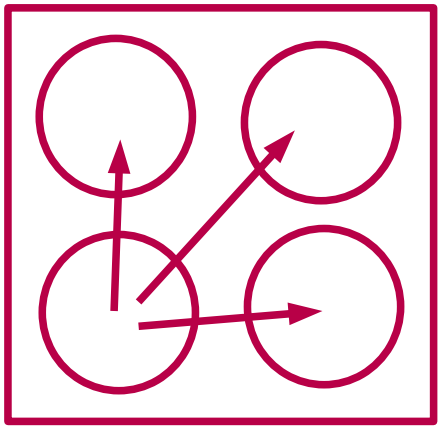
Modalità di gestione richieste HTTP



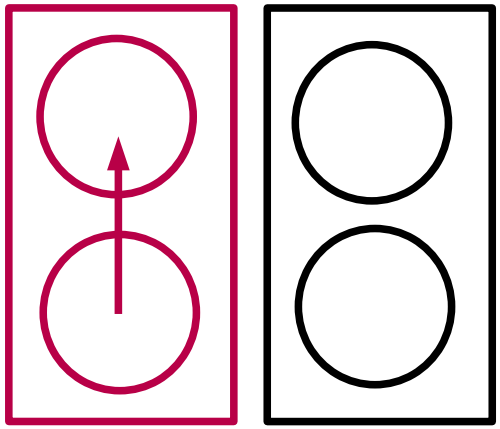
Due parole su sicurezza e prestazioni



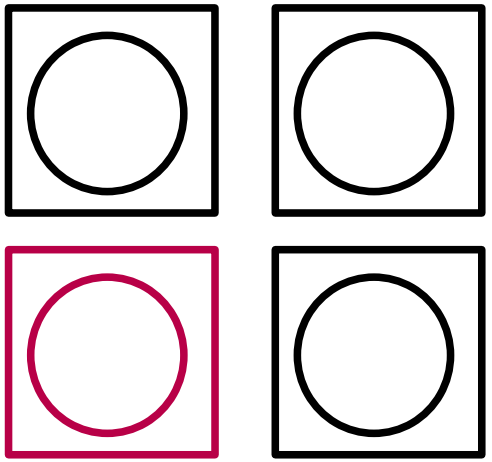
un processo molti thread



molti processi più thread per processo



molti processi un thread per processo



Nuova frontiera: gestione event driven

- **Gestione delle richieste event-driven**
  - Un solo processo
  - Un solo thread (o un thread per ogni core)
- **Ogni azione necessaria al servizio della richiesta è definita come una funzione**
  - Meccanismo che associa eventi a funzioni
  - Uso di puntatori
  - Completare l’esecuzione di una funzione è a sua volta un evento

Nuova frontiera: gestione event driven

- **Sistema che garantisce massime prestazioni**
  - Nessun overhead per context switch
  - Nessun overhead per passaggio dati
- **Comunemente usato in:**
  - Apache (event-based MPM)
  - Nginx
  - (Node.js)

Richiesta condizionale

- **Caching delle risorse**
  - Dopo un download le risorse sono memorizzate sul disco del client
- **Validazione della cache**
  - Non serve scaricare le risorse ogni volta
  - Non c’è un Time-to-live esplicito
  - Bisogna validare la cache
- **Metadati sulle risorse**
  - Data download
  - Etag (ID della versione)

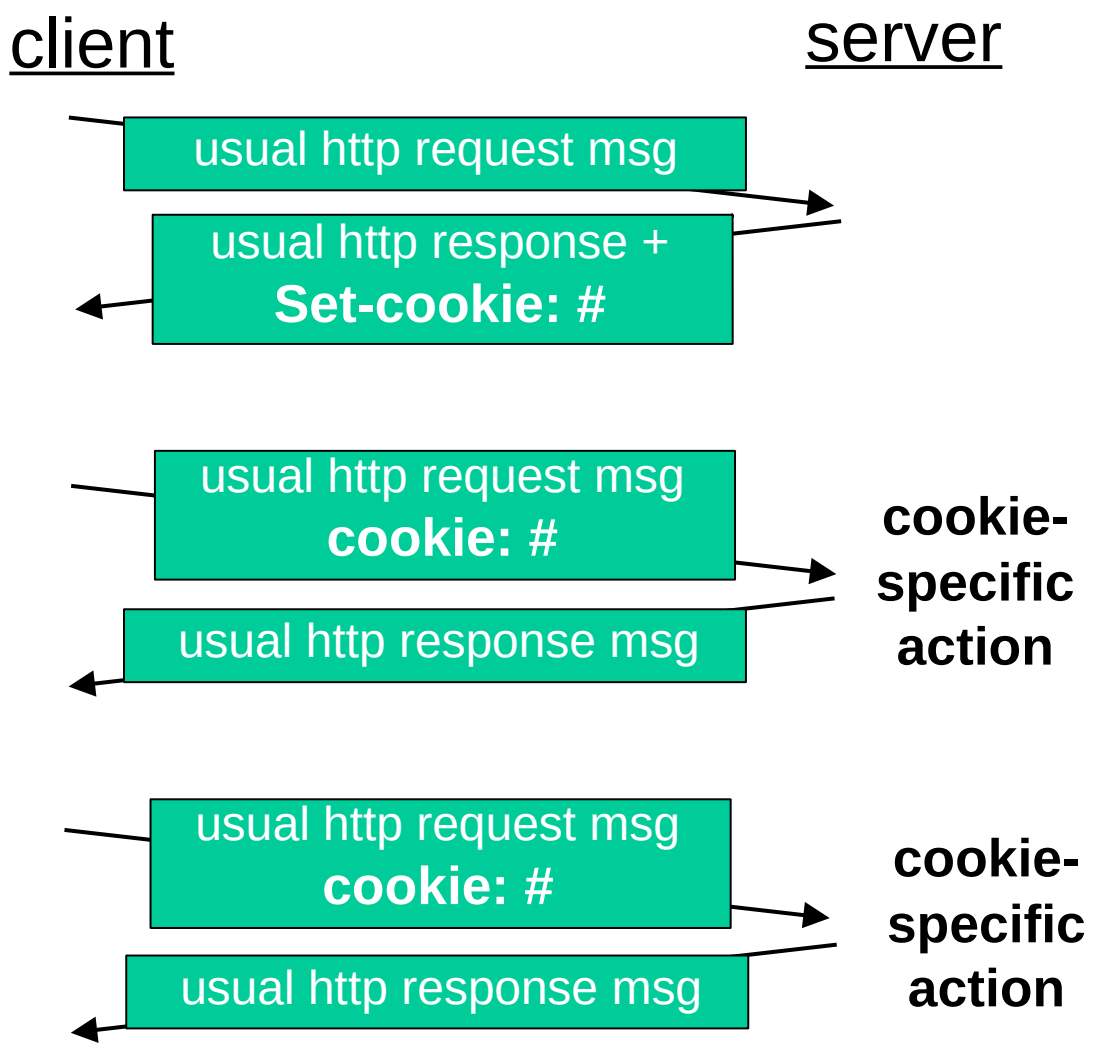
Richiesta condizionale

- **Richiesta di una risorsa**
  - Metodo GET
  - Header condizionali
- **Tipi di condizioni (Header HTTP)**
  - If-Modified-Since: <data-download>
  - If-None-Match: <etag-cache>
- **Se la risorsa in cache è aggiornata**
  - Response 304 – Not modified
  - Nessun corpo nella risposta
- **Se la risorsa in cache non è più valida**
  - Response 200 – OK
  - Corpo della risorsa

Interazione utente-server: cookie

- **Prima richiesta:**
  - Il server invia al client un ''cookie''
  - Header Set-cookie: #
- **Richieste successive**
  - Il client presenta il cookie
  - Header Cookie: #
- **Il server usa il cookie**
  - Identificare il client
  - Ricostruire la “sessione” per seguire le preferenze e la navigazione degli utenti
- **L’utente può cancellare i cookie (funzione dei browser)**

Interazione utente-server: cookie



Modulo 5
Cenni di
Web dinamico

Possibili utilizzi

- **Creazione di pagine personalizzate dinamicamente nel momento in cui vengono richieste in base a diversi parametri (es., dati client, ora/giorno, stato del sistema)**
- **Accesso ad informazioni gestite da server non HTTP, come i database ed altre applicazioni**
- **Interrogazioni a motori di ricerca**
- **Interazione personalizzata tra utente e server, che consente all'utente di effettuare accessi riservati, ricerche, acquisti e transazioni**

→ Il Web come interfaccia di servizi sempre più sofisticati

Risorse dinamiche

- **Alcune risorse Web non sono file multimediali, ma richiedono l'esecuzione di (uno o più) programmi**
- **L'aspetto importante è che l'utente non ha bisogno di rendersi conto che l'URL richiesto corrisponde ad un programma né che vi sia l'interazione con due o più server, in quanto il server HTTP trasmette il risultato dell'esecuzione e non il programma (≠ risorse attive)**

  → RICORDARE L'OBIETTIVO TRASPARENZA

Livelli logici di un servizio Web-based

- **In un servizio si possono approssimativamente distinguere, a livello logico, le seguenti componenti:**
  - Interfaccia utente, che rappresenta ciò che l'utente percepisce attraverso i propri sensi interagendo con l'applicazione
  - Logica di presentazione, che rappresenta quello che accade quando l'utente interagisce con l'interfaccia

Livelli logici di un servizio Web-based

- **… le seguenti componenti:**
  - Logica dell'applicazione, cioè le operatività cui è preposta l'applicazione (business logic)
  - Logica dei dati, cioè la gestione fisica dei dati (aggiornamenti e ricerche), compresa la loro validazione attraverso verifiche di completezza ed integrità
- **Questi livelli si trovano anche nelle applicazioni più moderne:**
  - Sviluppo con approccio MVC
  - Applicazioni multicanale
  - …
- **Possiamo riconoscere gli stessi livelli in contesti diversi**
  - Es. dentro a un Web Application Server

Importante

Non confondere i “livelli logici” con i “processi” che realizzano i livelli logici e con i “computer” che eseguono i processi.

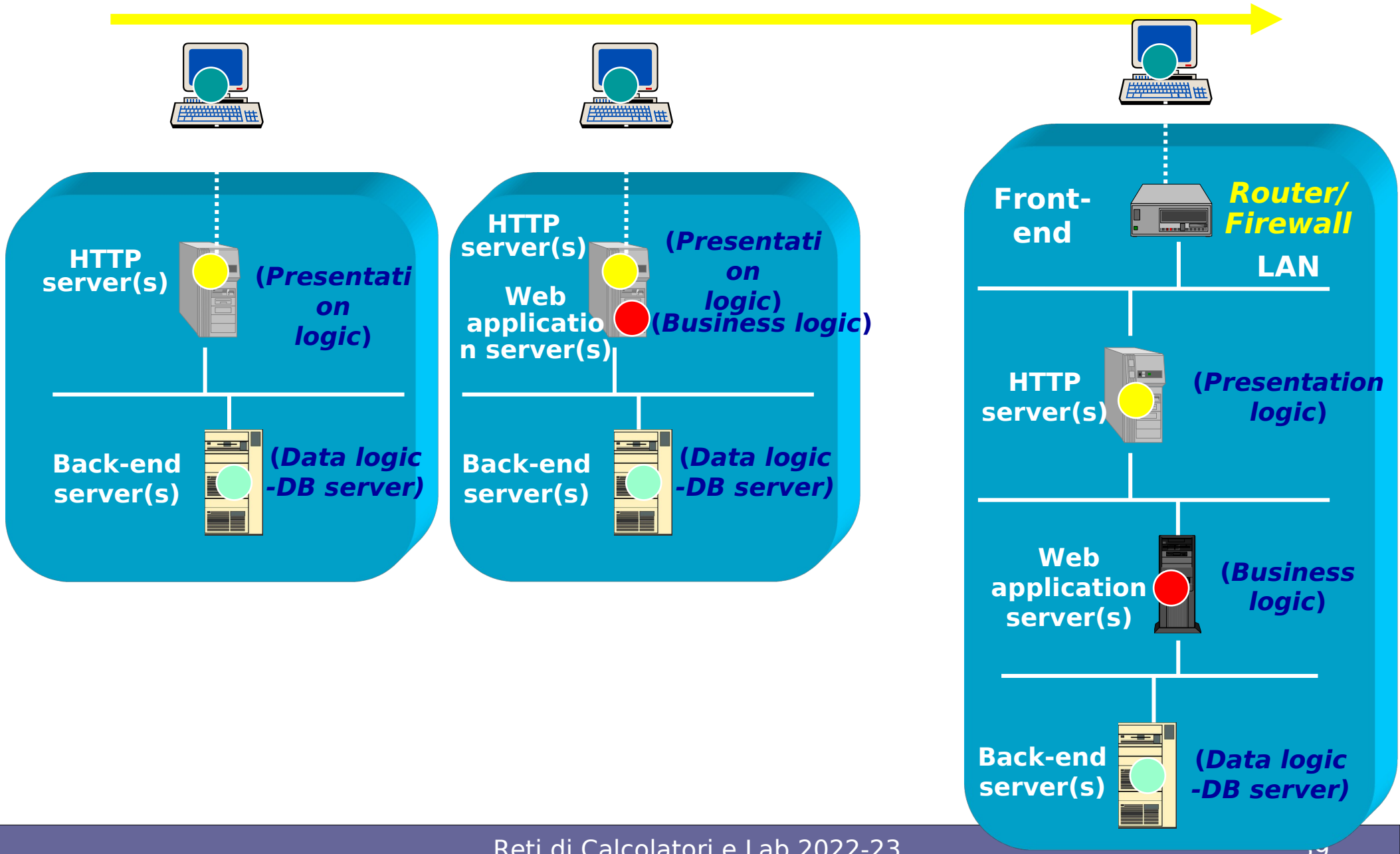
Quindi,

- Livelli logici
- Processi
- Computer (fisici/VM)

Esempio

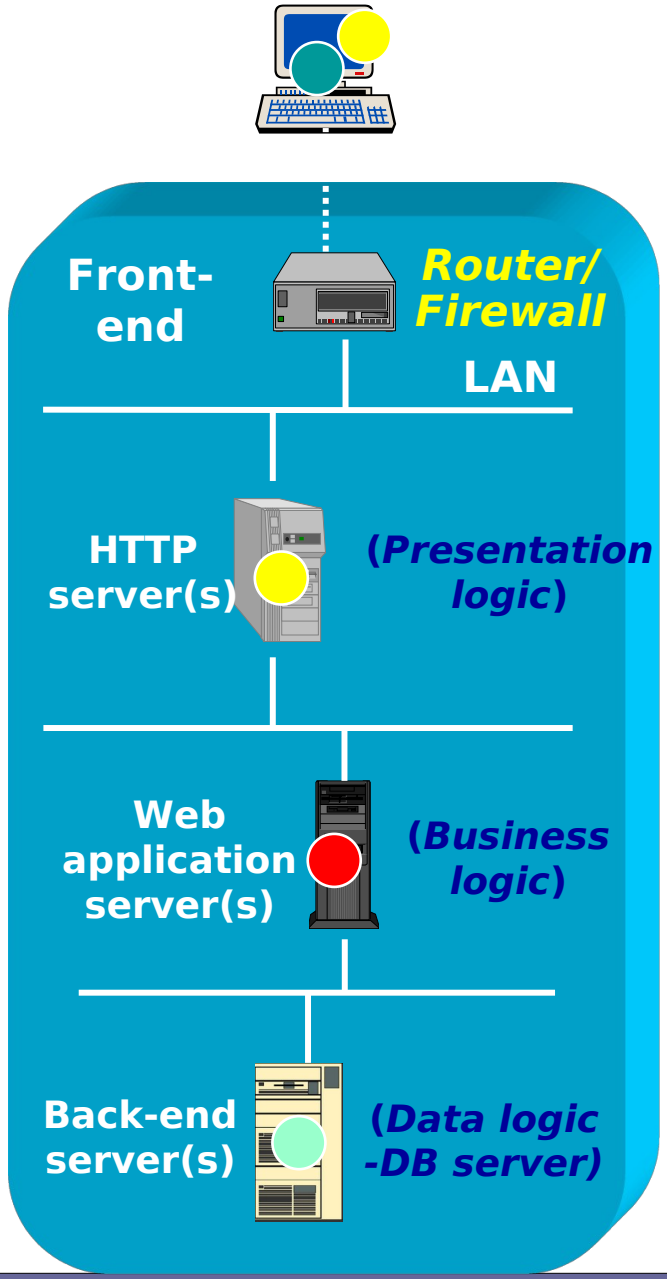
- **Quattro processi distinti**
- **Processo client: gestisce il livello “interfaccia utente”**
  - (Web Browser)
- **1° server: gestisce livello presentation**
  - Es: Server Web Apache HTTPd
- **2° server: gestisce livello application**
  - Es: Tomcat Web Application server
- **3° server: gestisce il livello data**
  - Es: DBMS PostgreSQL

Evoluzione architetture lato Web server

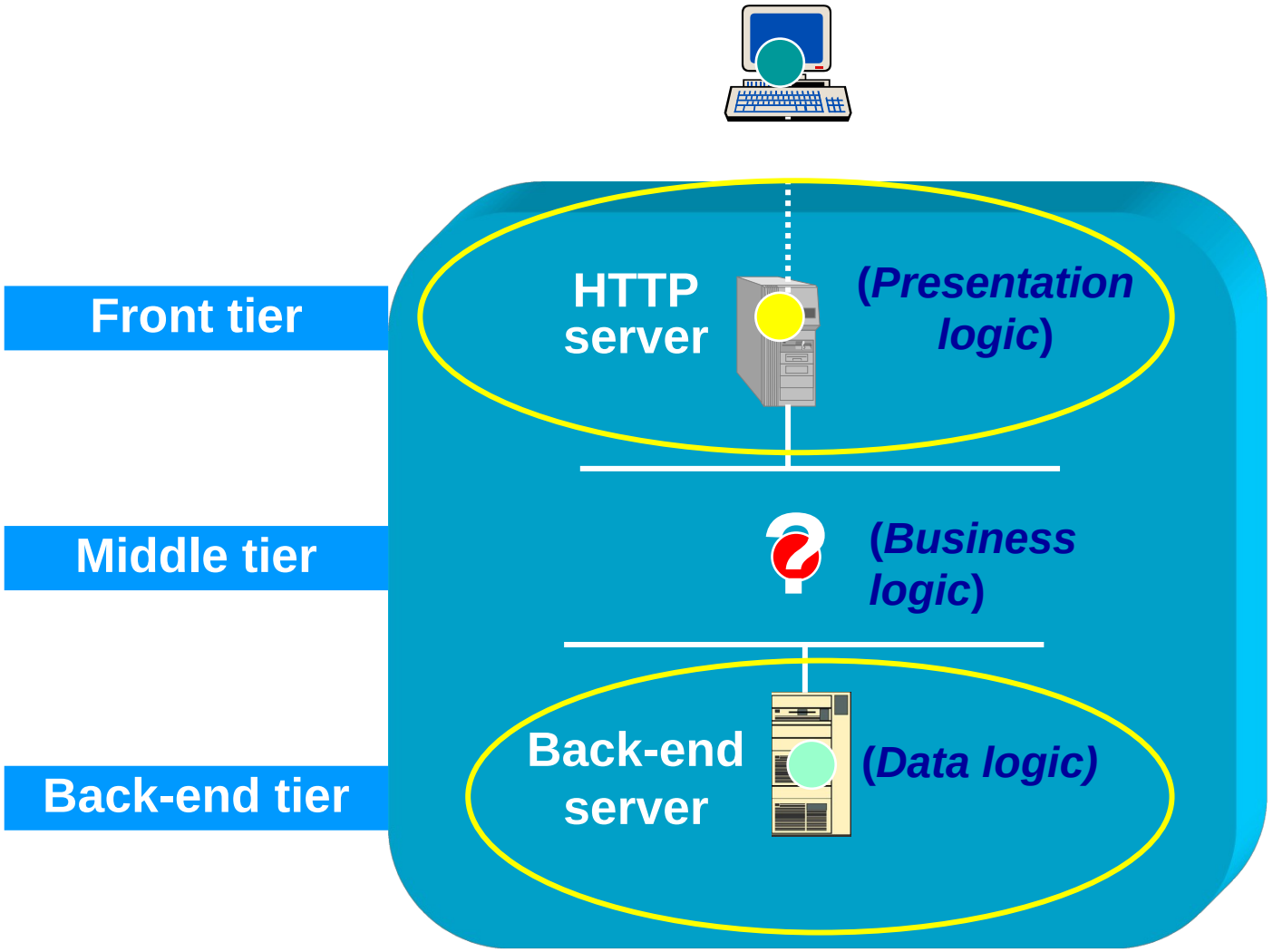


Una strada alternativa

- **Spostare parte della presentation logic sul client**
  - Migliora l'interattività
  - Riduce le latenze di risposta
  - Consente update asincroni della pagina
- **Approccio molto usato nelle tecnologie con forte dinamicità**
  - Tipicamente in contesti Web 2.0
  - Trend anche noto come “La vendetta delle applet”



Livelli definiti: Presentation & Data



Livello Business logic

- **Analizzare le tecnologie software**
- **Focus: Realizzazione di sistemi Web multi-livello (multi-tier)**
- **Obiettivo: realizzare servizi Web-based dinamici**

Tecnologie per middle tier

- **Tecnologie che prevedono processi esterni al server HTTP**
  - Versione base: un processo per ogni richiesta (es: CGI)
- **Linguaggi di scripting**
  - Integrati in Web server (es. PHP)
  - Processo esterno (es. Tomcat)
- **Tecnologie distribuite ad oggetti**

Primo passo di evoluzione

- **Se il problema è la scalabilità limitata**
- **Se il limite alla scalabilità è dovuto principalmente alla necessità di attivazione di un nuovo processo per ogni richiesta dinamica**

- **Si aumentino le prestazioni, mediante l'utilizzo di tecnologie che evitino la creazione di un nuovo processo**
- **→ Tecnologie di scripting**

Secondo passo di evoluzione (2)

- **Le tecnologie di scripting, che mirano principalmente all'aumento delle prestazioni, non soddisfano i nuovi requisiti richiesti dalle nuove applicazioni software**
- **La business logic complessa rende necessario una separazione dal processo server HTTP**
- **Attenzione: Scopi diversi. Non è un ritorno al CGI!**

- **Qual è la tecnologia che più risponde ai requisiti di:**
  - Modularità
  - Portabilità
  - Manutenibilità

Altra possibile tassonomia

- **Classificazione sulla base del focus nella fase di sviluppo**
  - Orientato alla pagina (es. PHP, JSP naive)
  - Object oriented (EJB naive)
  - MVC-oriented (framework specifici)
- **Classificazione sua base dell'implementazione**
  - Integrato nel Web server (es. PHP)
  - Processo esterno (es. CGI)
  - Server esterno (es. Fast CGI, Tomcat, etc...)

Driver dello sviluppo

- **Ridurre il tempo per rendere disponibili nuovi servizi**
  - Applicazioni mash-up
  - Composizione di servizi
- **Rinnovamento ed evoluzione dei servizi esistenti molto rapido**
  - Sistemi generalisti e facilmente modificabili
  - Sviluppo basato su framework molto flessibili

Modulo 6
Logging

Il logging

- **I logfile permettono di monitorare gli accessi ad un server Web**
  - Le informazioni che possono essere memorizzate nel logfile sono quelle che viaggiano all'interno dei messaggi di richiesta e risposta che il server scambia con il client usato dagli utenti
  - Generalmente i server Web permettono di definire quali campi dei messaggi devono essere memorizzati generando così dei logfile "custom" in modo da soddisfare al meglio le necessità dell'amministratore del sito Web

Logging nei Web server

- **Elemento chiave del logging: Valve**
- **Consente di definire:**
  - Quale file di log vogliamo scrivere
  - Che informazioni conservare nel file
  - Quale classe si occupa del logging
- **Struttura del file di log:**
  - Una riga per ogni richiesta
  - Le righe sono composte da record

Utilità dei log file

- **Monitorare lo stato del server**
- **Capacity planning**
- **Billing**
- **Attack detection** 

Esempio

127.0.0.1 - - [14/Oct/2002:18:00:16 +0200] "GET /icons/apache_pb.gif HTTP/1.1" 200 2326 "http://localhost/" "Mozilla/5.0 Galeon/1.2.6 (X11; Linux i686; U;) Gecko/20020913 Debian/1.2.6-2"

211.97.159.184 - - [14/Oct/2002:16:06:44 +0200] "GET /default.ida?
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
%u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u9090%u9090%u8190%u00c3%u0003%u8b00%u531b%u53ff%u0078%u0000%u00=a HTTP/1.0" 400 341 "-" "-"

Log analyzer

- **Esistono diversi strumenti che permettono di analizzare file di log. Alcuni sono generici, altri si focalizzano sui formati utilizzati dai più diffusi web server.**
- **Alcuni esempi sono:**
  - https://awstats.sourceforge.net/
  - https://webalizer.net/
- **Per esigenze limitate è possibile anche importare i file di log in programmi per la gestione di fogli di calcolo (es. Calc di OpenOffice)**