Lez01 - Elis - IT Architect

Architettura Web

Ing. Alberto Bucciero

Sommario

- Sistemi Informativi Distribuiti.
- Sistemi multi tier
- I protocolli alla base del Web
- Protocollo HTTP
- Pattern MVC

Cosa sono i Sistemi Distribuiti

- Sistemi nei quali le varie parti sono collocate su computer separati, eventualmente in luoghi diversi.

- Componenti, oggetti, dati sono installati dove possono svolgere al meglio il loro lavoro.

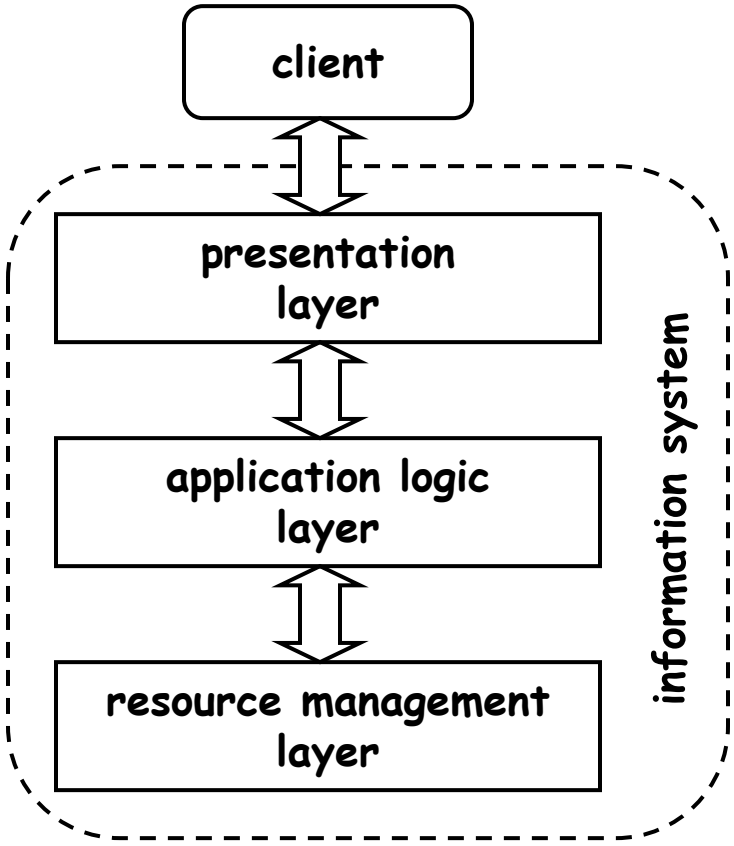
Caratteristiche dei Sistemi Informativi Distribuiti

- Logica e dati di un sistema distribuito possono essere acceduti da un client remoto in ogni momento e da qualsiasi luogo.
- Internet e il Web hanno dato un notevole impulso all'evoluzione dei Sistemi Informativi Distribuiti.
- Internet e le tecnologie ad esso associate hanno imposto degli standard (Protocolli, modalità di interazione etc.).
- La tecnologia ha dovuto adattarsi a questi standard e ha dovuto cercare soluzioni affidabili per la realizzazione dei Sistemi Informativi Distribuiti.

- Tiers e layers.
- Cosa sono?
- Conoscete la differenza?

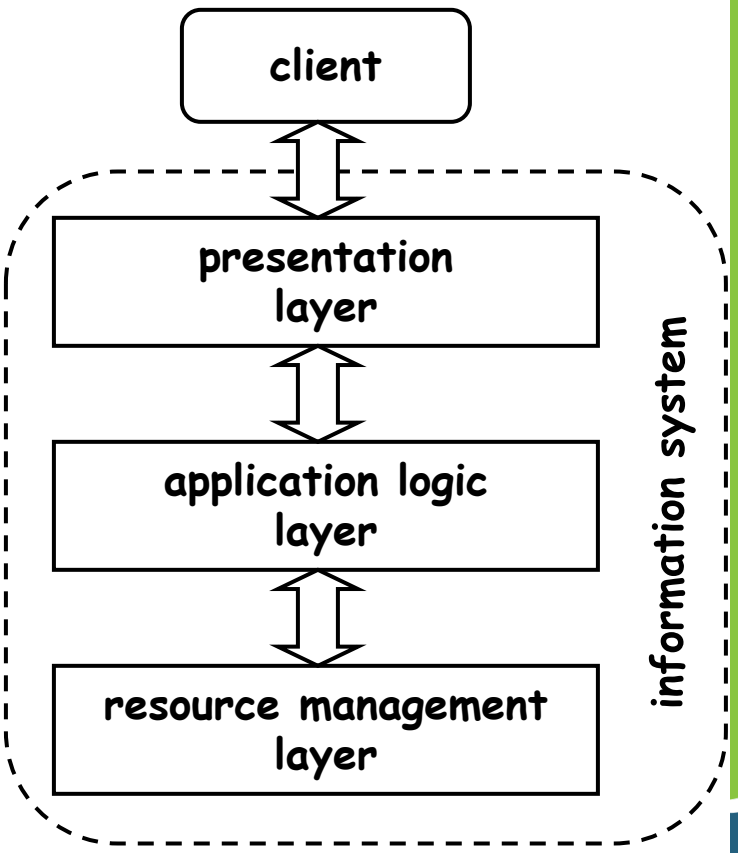
Sistemi Informativi (IS): layers

- Concettualmente un SI è organizzato in tre strati (è un tipico *pattern architetturale*)
- In alcuni casi questa separazione è puramente concettuale
- La separazione è fondamentale in fase di progettazione



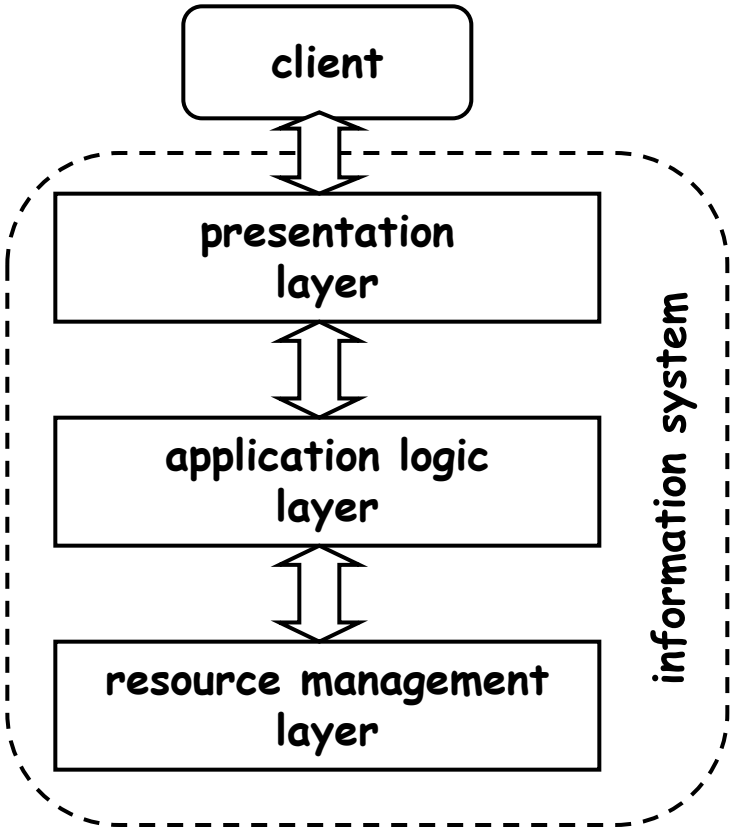
Sistemi Informativi (IS): layers – presentation logic

- *Il Client* è un programma che vuole realizzare una operazione sul sistema
- I client interagiscono con il sistema informativo attraverso lo strato di presentazione (**presentation layer**)
- Fornisce l'interfaccia grafica all'applicazione
- Tipica responsabilità è gestire l'interazione con l'utente:
  - Riceve l'input dell'utente
  - Invia dati al livello successivo (dopo eventuali verifiche)
  - Riceve dati dal livello successivo
  - Mostra risposte all'utente
- Idealmente non effettua elaborazioni ma si limita a gestire la presentazione dei dati
- Realizza l'obiettivo di separare la presentazione dalla logica di elaborazione



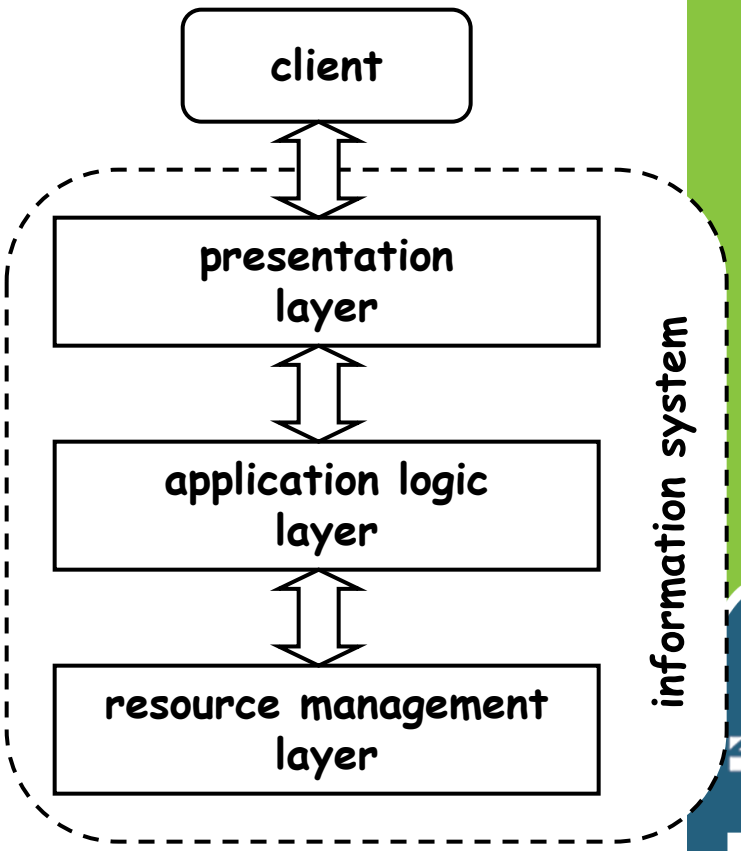
Sistemi Informativi (IS): layers – application logic

- I servizi che il SI mette a disposizione dei client rappresentano lo strato della logica applicativa (**application logic layer**)
- Fornisce regole e procedure di funzionamento dell'applicazione (business logic)
- Tipiche responsabilità:
  - Inviare e ricevere dati dal livello di presentazione
  - Validare l'input dell'utente
  - Gestire l'autenticazione dell'utente
  - Gestire l'autorizzazione dell'utente all' accesso a procedure e dati
  - Gestire le transazioni (accesso ai dati)
  - Gestire la concorrenza di vari utenti
  - Realizza l'obiettivo di disaccoppiare i client dai dati e di distribuire il carico di elaborazione tra i vari componenti



Sistemi Informativi (IS): layers – data layer

- Lo strato più basso, **resource manager**, si preoccupa della gestione (memorizzazione, indicizzazione, interrogazione) dei dati necessari alla logica applicativa.
- Tipicamente questo strato coincide con un DBMS
- Tipiche responsabilità:
  - Archiviazione, aggiornamento e cancellazione dei dati
  - Gestione dell'integrità dei dati
  - Funzioni di ricerca, ordinamento e filtro sui dati (opzionali)
  - Realizza l'obiettivo di garantire la persistenza delle informazioni in modo assolutamente indipendente dalla loro elaborazione e presentazione



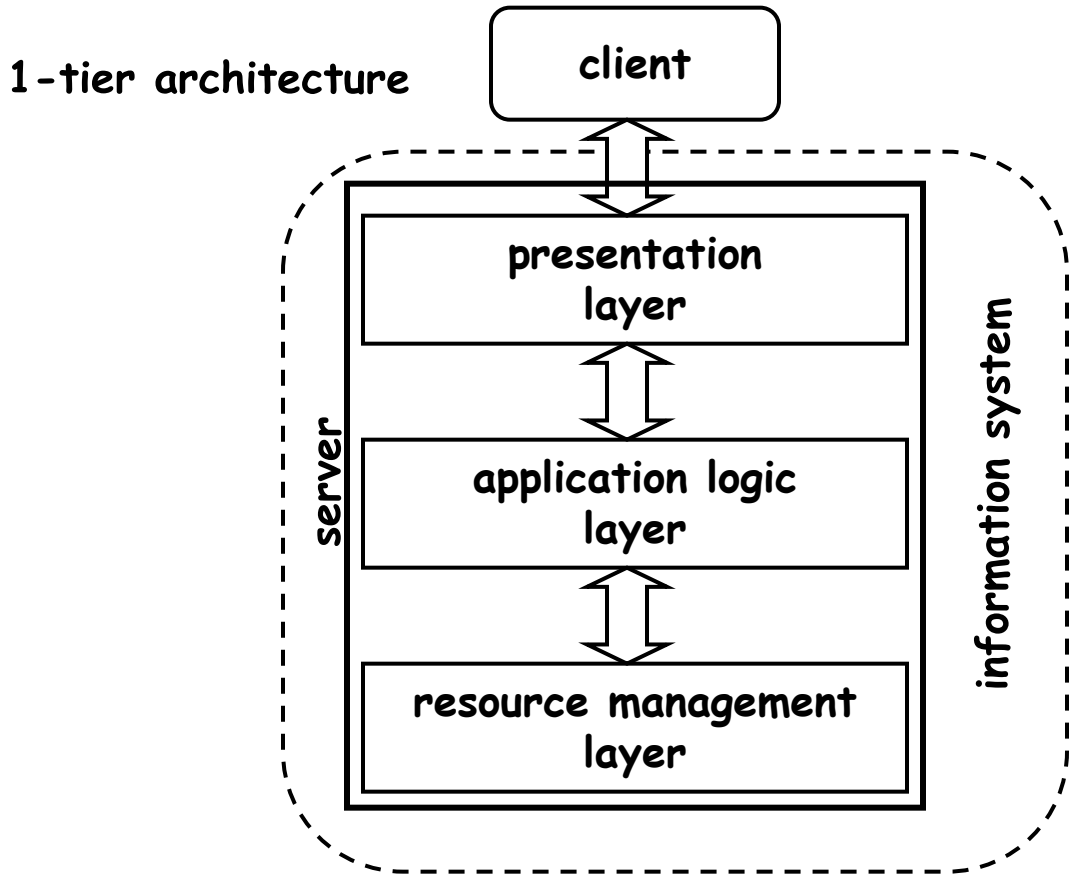
Sistemi Informativi: tiers

- Nella realizzazione concreta di un SI, i tre strati possono essere implementati e distribuiti in diversi modi
- In questo caso, ci riferiamo ad essi con il termine *tiers*
- In generale la differenza tra layer e tier consiste nel fatto che:
  - mentre i primi fanno riferimento ad una scomposizione concettuale,
  - i secondi corrispondono ad una scomposizione logica che può consentire una scomposizione fisica di ogni strato su una macchina diversa

Tiers e Layers

- Un moderno sistema informativo deve essere modulare
  - In particolare oggi possiamo avere molti diversi tipi di client che richiedono i servizi offerti dal sistema; inoltre l'interazione tra il client e il sistema può avvenire attraverso canali con caratteristiche diverse e variabili
- Nel seguito presentiamo (schematicamente) l'evoluzione dell'architettura dei Sistemi informativi

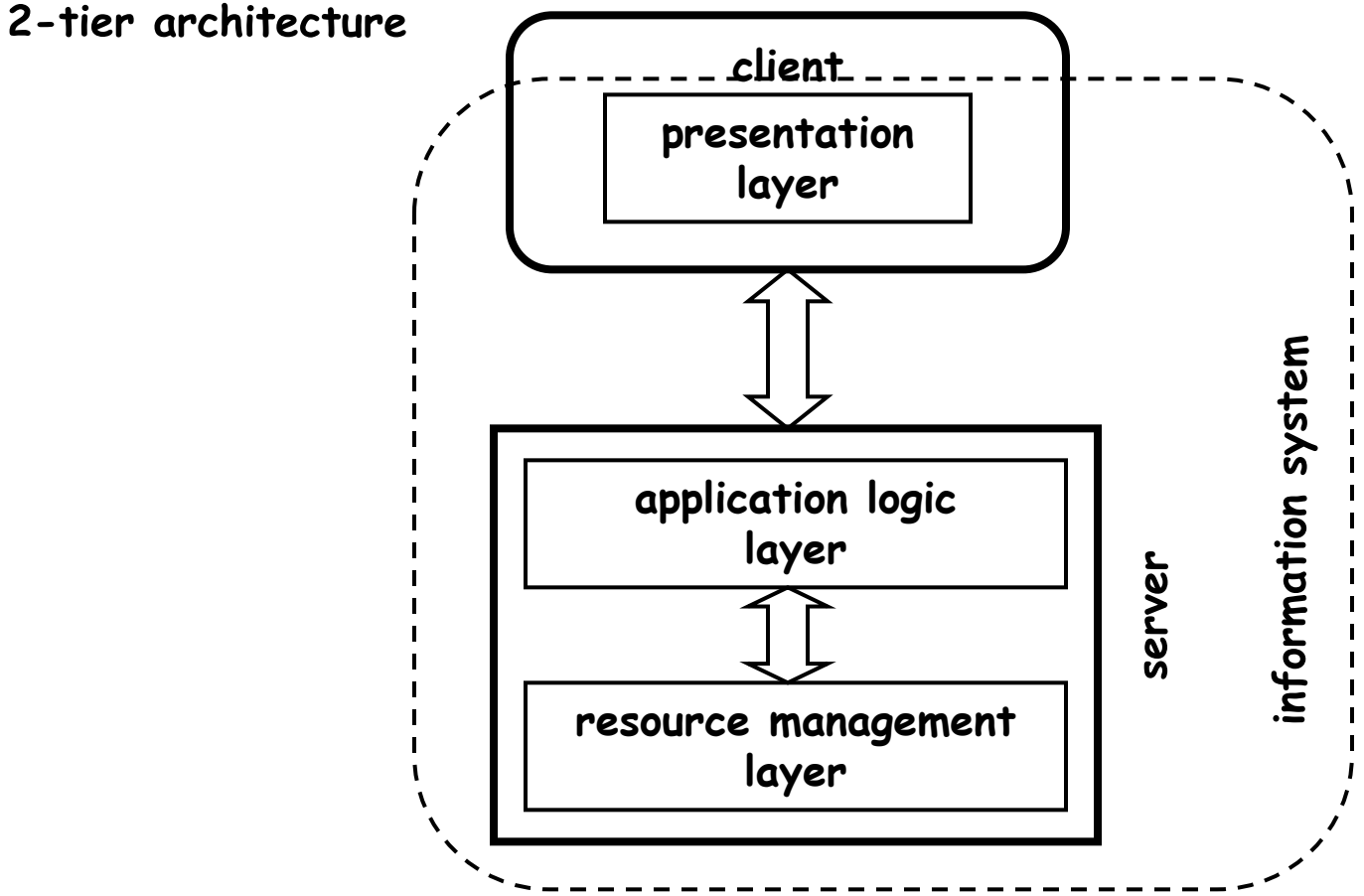
Un tier: tutto centralizzato



One tier: tutto centralizzato

- Tutti e tre gli strati sono realizzati come una struttura monolitica
- Utenti e programmi accedono al sistema attraverso terminali (a carattere) "stupidi": ciò che viene presentato e come deve apparire è controllato direttamente dal server
- Questa è la tipica architettura dei mainframe (ancora molto diffusi in grosse organizzazione come banche, assicurazioni, compagnie aeree)
- Vantaggi:
  - Tutto avviene all'interno del sistema
  - La gestione e il controllo delle risorse può essere più semplice
  - La progettazione può essere ottimizzata
- Svantaggi:
  - Non esiste un punto di ingresso dall'esterno, quindi è molto difficile integrare il sistema con altri sistemi esterni (*screen scrapers*)
  - La manutenzione e l'evoluzione del sistema è estremamente costosa

Two tier: client/server

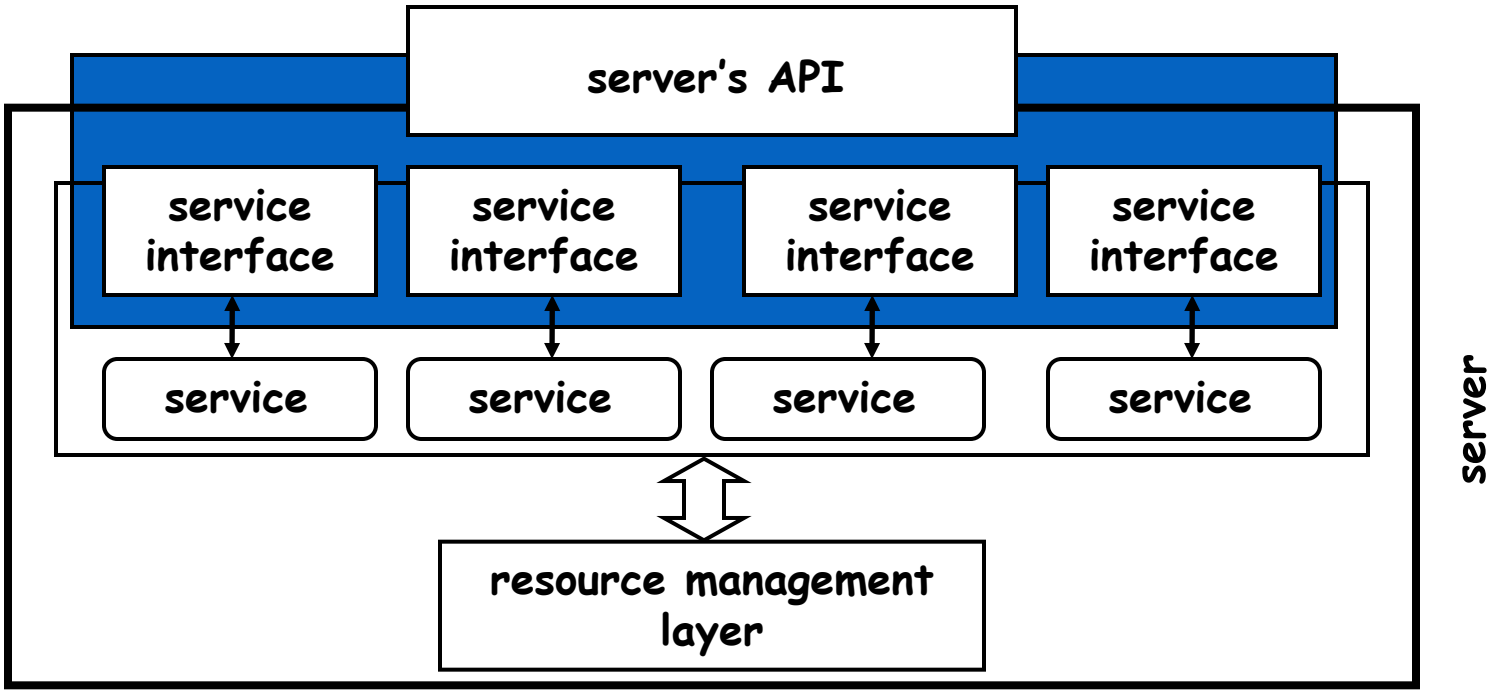


Two tier: client/server

- L'evoluzione della potenza dell'hardware e della affidabilità delle reti, ha permesso di spostare lo strato di presentazione sul client.
- Tipica soluzione: lo strato di presentazione è realizzato con una applicazione Visual-Basic su un pc; gli altri due strati rimangono su un server
- Vantaggi
  - I client sono indipendenti fra loro: possiamo avere molti strati di presentazione a seconda delle necessità
  - Si possono sfruttare le risorse di calcolo del client per realizzare presentazioni sofisticate, riducendo il carico sul server
  - Il resource manager vede e interagisce solo con lo strato sovrastante (application logic): si possono ottenere miglioramenti di prestazioni riducendo le sessioni e le connessioni da mantenere
  - Nella progettazione del server, ignorando i problemi di presentazione c'è spazio per ottimizzazioni
  - Il controllo e la manutenzione sono ancora relativamente semplici
- Un inciso importante
  - Si introduce il concetto di API (Application Program Interface)

Il concetto di API

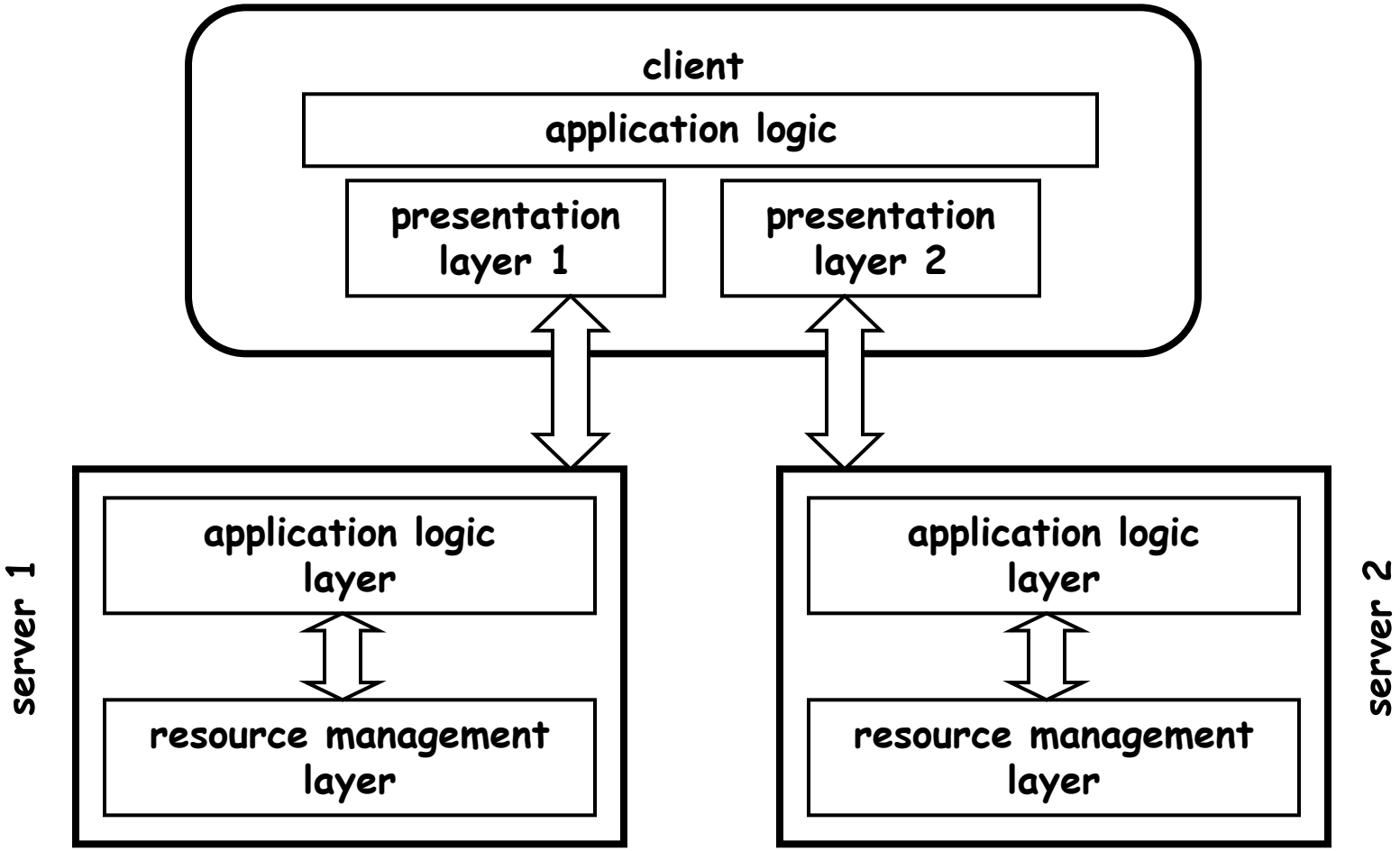
- Le architetture Client/server introducono il concetto di **servizio** (il client invoca un servizio implementato dal server)
- Con questa nozione, viene introdotto anche il concetto di **service interface** (specifica come i client possono invocare un dato servizio)
- **Application Program Interface** (API) descrive come interagire con il server dall'esterno per invocarne i servizi



Limiti delle architetture client/server

- Con l'aumentare della affidabilità e della banda delle reti nascono opportunità di integrazioni (di sistemi differenti verso un’interfaccia unificata)
- Ma l'integrazione, nelle architetture client/server viene realizzata sul tier corrispondente al client. Questa soluzione però non è scalabile.

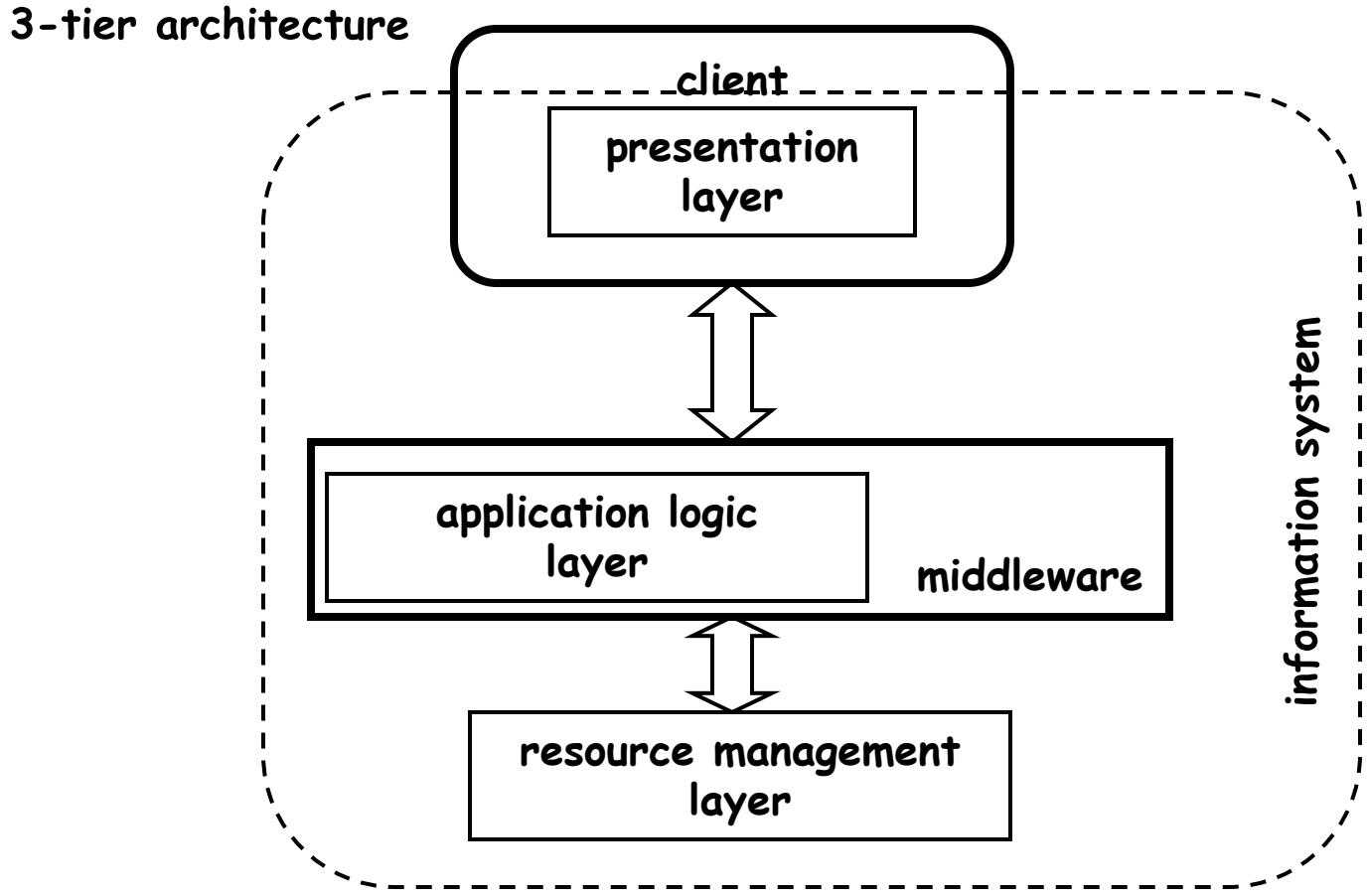
Limiti delle architetture client/server



Limiti delle architetture client/server

- Se il client deve accedere a servizi offerti da due o più server, una architetture a 2-tier comporta molti problemi:
  - Non c'è una logica comune: il client diventa il punto dove viene realizzata l'integrazione
  - La responsabilità di affrontare la possibile eterogeneità dei sistemi è affidata al client
  - Il client diventa quindi responsabile di sapere dove stanno le informazioni, come accedervi, e, aspetto più inquietante, come mantenerne la consistenza delle informazioni
  - Portabilità
  - Le prestazioni del client decadono
- Se da una parte si intravede l'opportunità di creare nuove applicazioni integrando i servizi offerti dai server, dall'altra non c'è l'architettura adatta a sostenere questo tipo di sviluppo

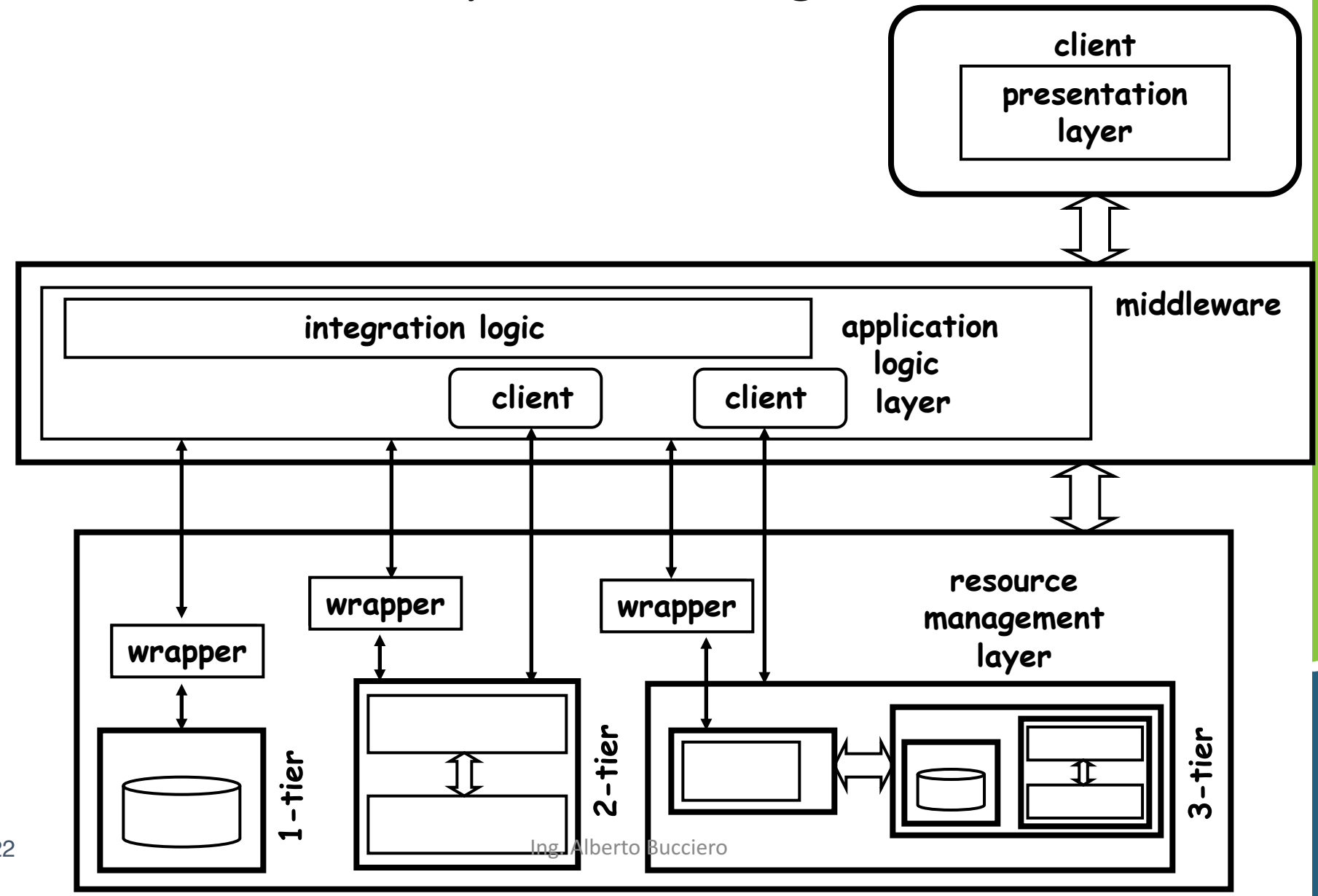
Architetture *Three tier*



Architetture *Three tier*

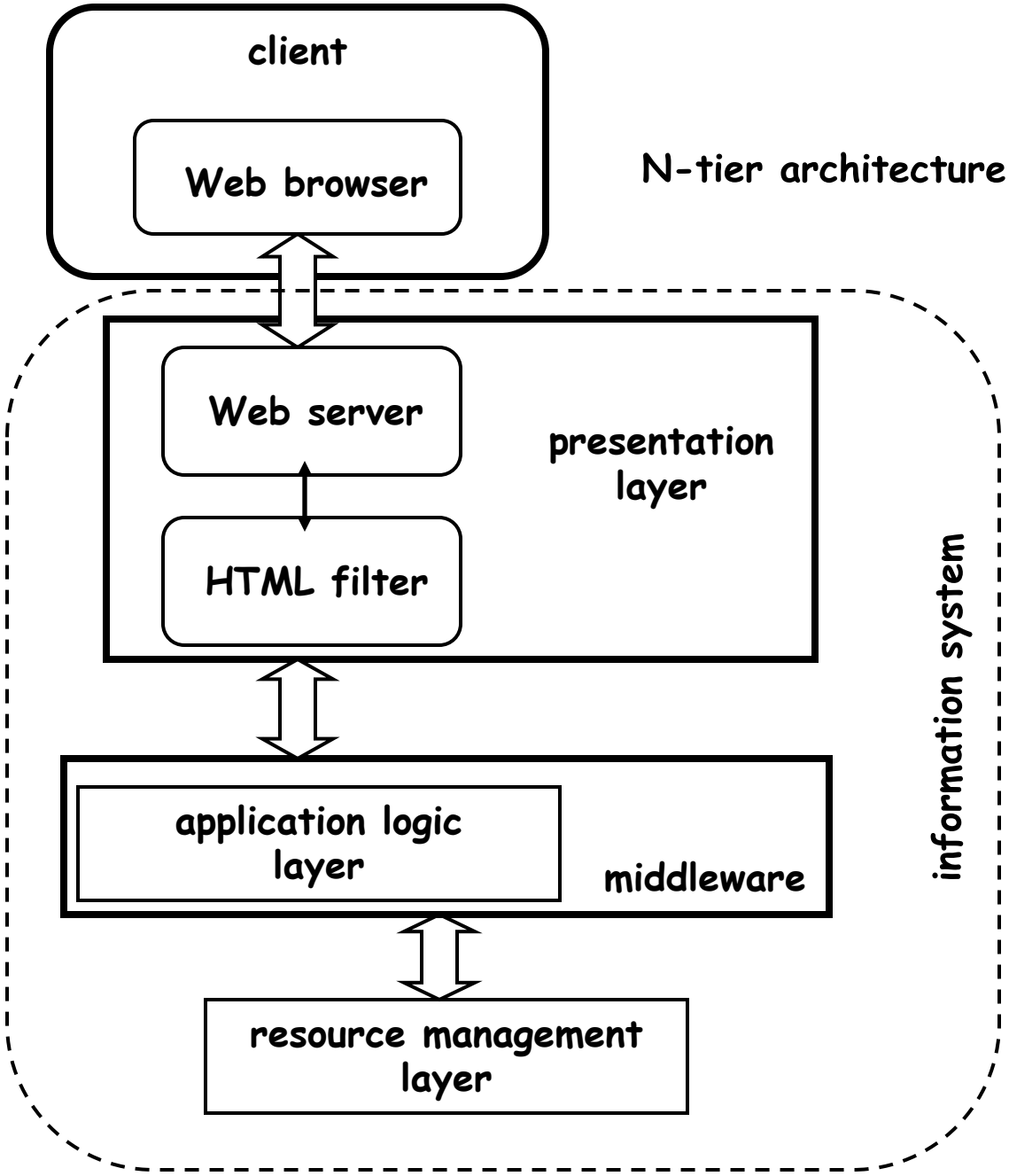
- Le architetture 3-tier nascono come soluzione a queste nuove esigenze
- L'idea è quella di introdurre un tier entro il quale viene sviluppata la logica applicativa, che può essere risultato della integrazione di più servizi
- In generale le architetture 3-tier sono molto più complesse e variegate rispetto alle architetture c/s
- In astratto possiamo dire che in queste architetture c'è una chiara separazione tra ciascuno dei tre layer
  - Lo strato di presentazione risiede sul client
  - Lo strato della logica applicativa risiede sul tier intermedio. Questo offre una infrastruttura di supporto allo sviluppo della logica intesa come integrazione di più servizi, e prende il nome di *middleware*
  - Lo strato inferiore è formato dai server (fornitori di servizi) che la logica applicativa cerca di integrare.

Middleware e System integration



Un'ulteriore evoluzione N-tier

- Nell'architettura 3-tier i servizi offerti dallo strato inferiore possono essere a loro volta sistemi a 1-tier, 2-tier, 3-tier …
- Il Web si impone come canale di accesso; un server Web deve quindi essere incorporato nello strato di presentazione
- Poiché è piuttosto complesso sviluppare uno strato di presentazione che include il server Web, questo viene considerato un vero e proprio tier aggiuntivo
- In questi sistemi il client è un browser e il layer di presentazione è distribuito tra il Web browser, il server Web e il software che usa i dati forniti dalla logica applicativa per preparare le pagine HTML

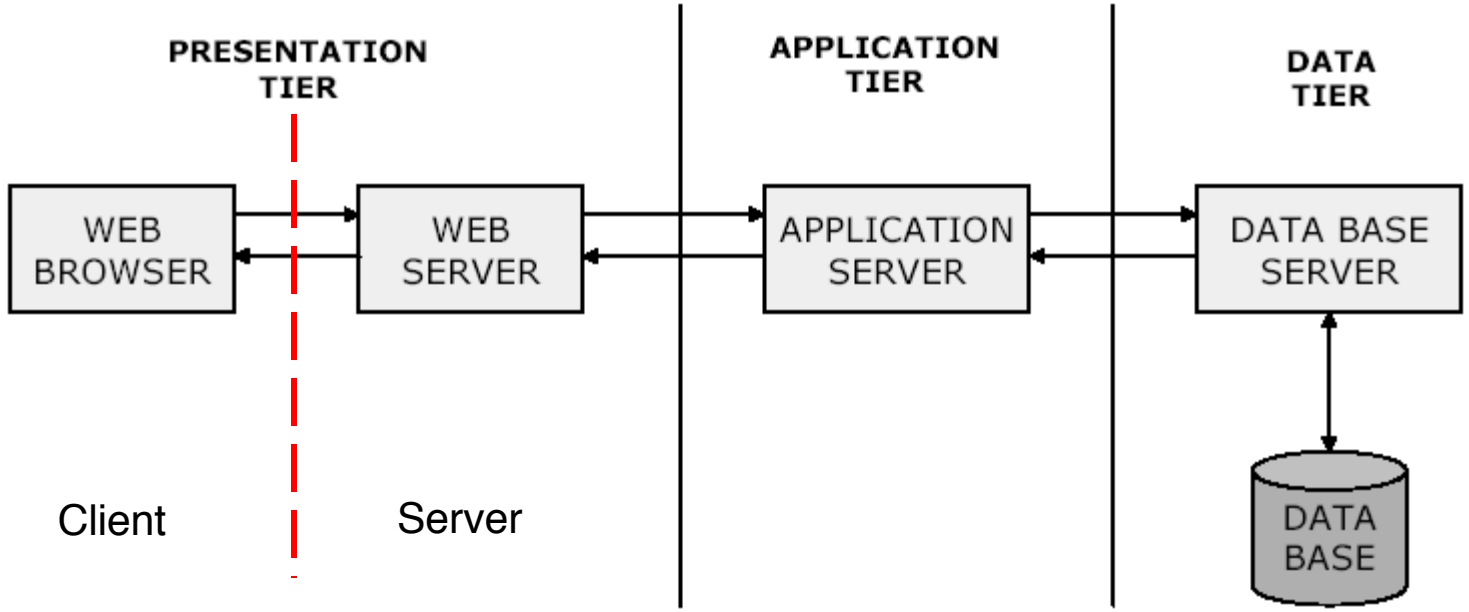

client
Web browser
N-tier architecture
Web server
presentation layer
HTML filter
information system
application logic layer
middleware
resource management layer

N-tier: Sistemi informativi su Web

- Le architetture N-tier sono il risultato della connessione di tanti sistemi 3-tier, con l'aggiunta di un layer specifico per consentire ai client di accedere al sistema attraverso un Web server
- Inizialmente lo strato per l'accesso Web era esterno al sistema; oggi, sta iniziando ad essere incorporato in uno strato di presentazione che risiede sul lato server
- La aggiunta dello strato Web ha portato alla nozione di *application servers*, che viene usata per riferirsi a piattaforme di middleware che supportano l'accesso attraverso il Web

Server

- Rappresentazione alternativa delle architetture web n-tier



Soluzioni applicative

- Il modello client-server nell'architettura a tre livelli si realizza tipicamente con:
- Presentation-tier:
  - Web Browser
  - Web Server con:
    - HTML
    - linguaggi di scripting lato client (Javascript)
    - Java Applet
- Application-tier:
  - Application server con
    - linguaggi di programmazione (Java, Perl,C++, C#)
    - linguaggi di scripting lato server (PHP, JSP, ASP)
- Data-tier:
  - Data Base Server (linguaggio SQL) con dati:
    - in forma relazionale o ad oggetti o NOSQL

La programmazione lato client (1)

- Insieme di tecniche che permettono di elaborare dinamicamente lato client le risorse fornite dal server
- Javascript con associate librerie DOM (Document Object Model)
  - Caratteristiche:
    - Parzialmente orientato agli oggetti e simile a Java nella sintassi
    - Standard e compreso da tutti i browser
  - Utilizzato per:
    - Manipolare le pagine attraverso l'accesso ai componenti DOM
    - Validare l'input lato client nei form HTML
    - Realizzare effetti grafici e animazioni
- AJAX : Fornisce nuovi metodi per l'utilizzo di JavaScript, linguaggi lato server (ASP.Net o PHP) e altri linguaggi al fine di migliorare l'esperienza dell'utente.
- Adobe Flash Player onnipresente piattaforma client-side, ma oramai obsoleta.

La programmazione lato client (1)

- **Java Applet**
  - Caratteristiche:
    - Programmi in bytecode scaricati dal server web (come una qualsiasi risorsa) ed eseguiti dalla JVM inclusa nel browser
    - Nessun accesso alle risorse del computer client
    - Comunicazione solo con il server di partenza
  - Utilizzato per:
    - Animazioni nelle pagine web
    - Funzioni di grafica evoluta
    - Realizzare in genere funzioni non supportate da HTML + Javascript senza dover ricorrere a programmazione lato server
- **Java Web Start** (noto anche come **JavaWS**, **javaws** o JAWS) è un framework sviluppato da Sun Microsystems (ora Oracle), che permette agli utenti di scaricare ed avviare applicazioni software per **Java** Platform direttamente da Internet utilizzando un browser **web**.

La programmazione lato server

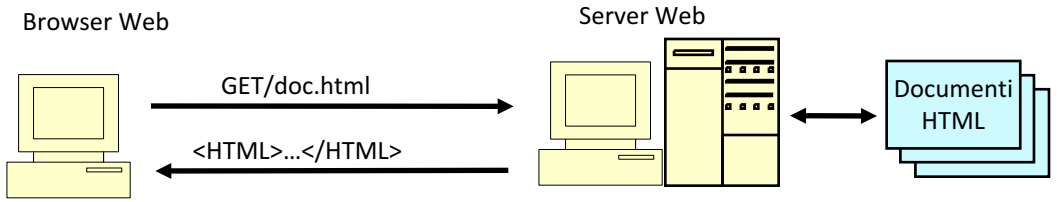
- Insieme di tecniche che permettono di realizzare dinamicamente lato server le risorse da restituire al client
- Tali tecniche si appoggiano su vari linguaggi di programmazione e di scripting
- Più comunemente utilizzate:
  - Common gateway Interface (CGI)
  - Active Server Pages (ASP) e (ASPX)
  - Java Server Pages (JSP)
  - Personal Home Page (PHP)
  - Java Servlet
  - Piattaforma Java EE (Java Enterprise)
  - Piattaforma .NET

Cos'è il WWW

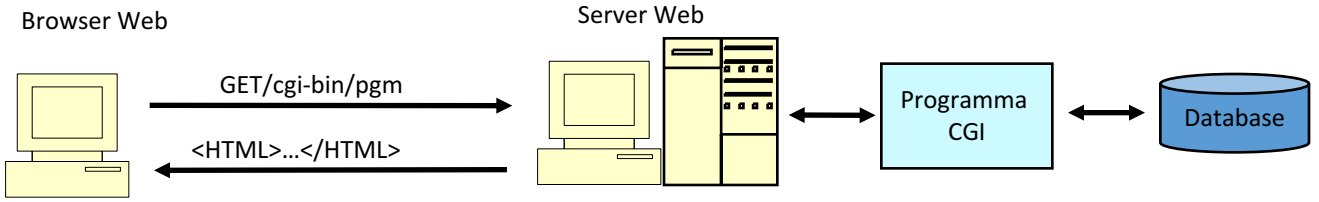
- Il World Wide Web è un sistema per la presentazione a schermo di documenti multimediali, e per l'utilizzo di link ipertestuali per la navigazione.
- Il sistema è distribuito e scalato su tutta Internet, ed è basato su alcuni semplici concetti:
  - Il **client** o **browser** è un visualizzatore di documenti ipertestuali e multimediali. Può visualizzare testi, immagini e semplici interfacce grafiche, ma non permette di editare documenti.
  - Il **server** è un semplice meccanismo di accesso a risorse locali (file o record di database, ecc.) in grado di trasmettere via socket TCP documenti individuati da un identificatore univoco
  - Il server può collegarsi ad applicazioni server-side (p.e. tramite protocollo CGI) ed agire da tramite tra il browser e l'applicazione facendo del browser l'interfaccia dell'applicazione.

Un po' di storia

- Il primo modello operativo del Web prevedeva la presenza di un server che non faceva altro che servire documenti su richiesta.



- Ben presto divenne chiaro che una richiesta HTTP potesse venire interpretata come una query a un db e che i suoi risultati potessero essere usati per costruire in modo dinamico un documento HTML.



Cos'è una CGI

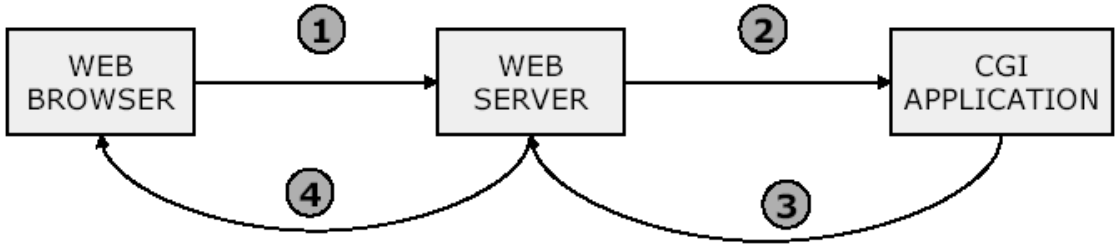
- Una CGI è un programma che viene richiamato dal server Web per rispondere a determinati tipi di richieste, normalmente richieste di documenti in una particolare directory o di file con l'estensione .cgi. I parametri della richiesta vengono passati come coppie chiave/valore
- Il programa legge i parametri, svolge le operazioni specificate dall'applicazione (tipicamente un accesso al db) e genera in risposta un documento HTML inviato attraverso l'HTTP come se si trattasse di un normale documento statico.

PROBLEMI:

- Nelle CGI, ciascuna richiesta HTTP genera tipicamente un nuovo processo nel server che richiede memoria, tempo di cpu ed occupa altre risorse dell'host (FastCGI però gestisce tutte le richieste come un unico processo).
- Inoltre le CGI hanno un'interfaccia limitata che non supporta servizi di alto livello come sicurezza, gestione dello stato, load balancing e scalabilità.

CGI

- Realizzata con diversi linguaggi (più usato il Perl)
  1. Si chiama un programma CGI sottomettendo i dati di una form HTML
  2. Il server web inoltra i dati all'applicazione CGI
  3. L'applicazione CGI restituisce la risorsa elaborata (una pagina HTML) al server web
  4. Il server web restituisce la risorsa al browser



ASP

- Tecnologia proprietaria sviluppata da Microsoft
- Le pagine ASP sono pagine HTML o XHTML e integrano all'interno porzioni di codice scritto in VBScript, codice che viene eseguito sul server
- Gli elementi di scripting sono identificati da <% %>
- Supportate solamente dai server Internet Information Server (IIS) di Microsoft

PHP

- Linguaggio ibrido tra C, Perl e JavaScript
- Ottimo supporto per vari sistemi di Data Base
- Supporto anche per servizi che comunicano con altri protocolli (oltre ad HTTP), cioè: SMTP, POP3, IMAP e NNTP
- Le pagine PHP integrano all'interno porzioni di codice (in PHP) che viene eseguito sul server
- I comandi sono marcati con <?php ?>

Piattaforma .NET

- Piattaforma Microsoft che fornisce tutti gli elementi per sviluppare ed eseguire applicazioni web (e non) basate sul linguaggio XML per lo scambio dati
- Fornisce:
  - Strumenti di sviluppo
  - Librerie e interfacce di programmazione (API)
  - Supporto per vari linguaggi tra cui C#, C++, VB.NET, ASP.NET
  - Infrastruttura di server per l'esecuzione delle applicazioni
  - Servizi di base già predisposti

Tecnologie Java

- JSP
  - Le pagine JSP sono pagine HTML o XHTML che integrano all'interno porzioni di codice Java che viene eseguito sul server
  - Gli elementi di scripting sono identificati da <% %>
  - Evoluzioni recenti per meglio separare l'HTML dalla parte di scripting: introduzione delle Tag Library
- Servlet
  - Componenti Java che forniscono una struttura generale per lo sviluppo di servizi basati sul modello richiesta-risposta
  - Diversamente da CGI possono gestire richieste multiple in modo concorrente
- Java EE
  - Piattaforma completa che include una serie di componenti (tra cui JSP, Servlet, EJB) per lo sviluppo e l'esecuzione di applicazioni web (e non)
  - altamente scalabili e fornite di gestione della **sicurezza**, delle **transazioni** e della **concorrenza**.

I protocolli del WWW

- Alla base di WWW ci sono i seguenti protocolli:
  - Uno **standard per identificare** in maniera generale **risorse di rete** e per poterle specificare all'interno di documenti ipertestuali (chiamato **URI** es. http://alice:secret@www.example.com:80/path/to/document?foo=bar [<scheme>://<authority><path>?<query>]).
  - Un **protocollo di comunicazione state-less e client-server** per l'accesso a documenti ipertestuali via rete (chiamato HTTP).
  - Un linguaggio per la realizzazione di documenti ipertestuali (chiamato HTML) basato su SGML e incentrato sulla possibilità di realizzare connessioni ipertestuali in linea nella descrizione strutturale del documento.

Evoluzioni del WWW (1)

- **Inclusione di oggetti**: Mosaic introdusse il supporto per immagini in-line, e Netscape introdusse poi i plug-in per inserire oggetti di tipi diversi nel documento del browser, ed infine le applet Java. IE ha generalizzato le possibilità offerte da COM, realizzando il protocollo proprietario ActiveX.
- **Scripting**: Netscape introdusse LiveScript, poi ribattezzato Javascript, per realizzare semplici applicazioni client-side con linguaggi di scripting appositi. IE rispose con Jscript e Vbscript.
- **Stili**: L'uso di trucchi per forzare HTML a rese grafiche insolite ha portato a creare linguaggi appositi per gestire gli aspetti di visualizzazione del documento. CSS permette di controllare le caratteristiche dei documenti HTML. XSL si occupa di documenti XML.
- **Server-side includes**: La possibilità di inserire all'interno del documento HTML memorizzato sul server delle istruzioni eseguite dal server stesso prima di spedire il risultato hanno permesso notevoli sofisticazioni nel contenuto (data di oggi, utente registrato, numero di visitatori, piccole computazioni, ecc.)

Evoluzioni del WWW (2)

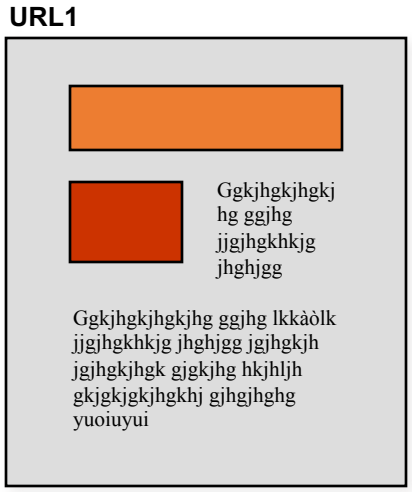
- **Gestione delle transazioni**: meccanismi per la gestione dello stato sono stati introdotti prima da Netscape, e poi standardizzati (**cookies**). Meccanismi di accesso in scrittura e cooperazione a risorse WWW vengono studiati in questo periodo (WebDAV).
- **Siti web dinamici**: I server-side includes e le applicazioni CGI evolvono e si fondono, diventano veri e propri linguaggi o ambienti di programmazione: Perl prima, poi ASP (Javascript e Visual Basic), poi PHP, Python, ecc. Con l'arrivo delle librerie di accesso ai database (ODBC, JDBC) nasce l'architettura a tre livelli (user-interface, application logic, data storage). Il browser diventa l'interfaccia di eccellenza per applicazioni gestionali distribuite.
- **Strutturazione dei documenti**: i limiti di HTML non erano soltanto nella visualizzazione, ma anche nella strutturazione. XML permette di definire linguaggi di markup più adatti ai singoli task.

Il web oggi

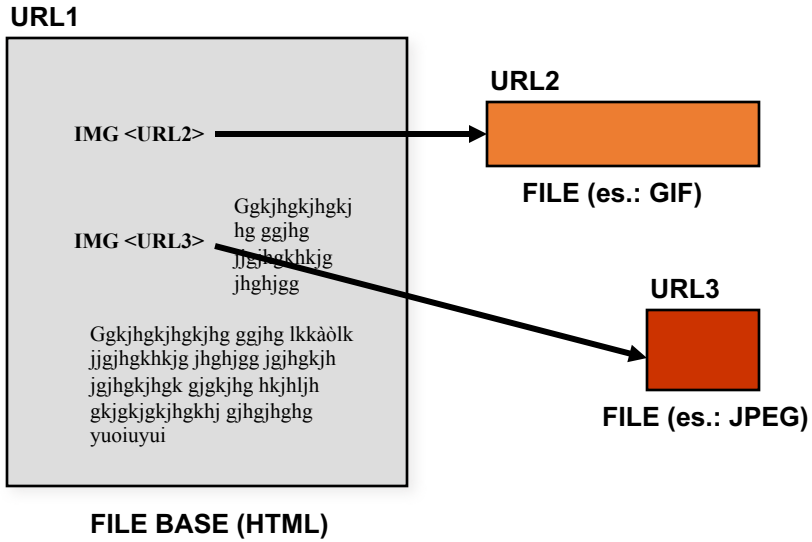
- Accessibilità
  - la possibilità di accedere alle informazioni ed ai servizi da parte di persone limitate da disabilità fisiche, linguistiche, tecnologiche
- Mobilità
  - La possibilità di accedere alle informazioni ed ai servizi da qualunque parte del mondo, con hardware diverso (e più limitato) del classico PC con schemro a colori: palmari, cellulari, car stereo, home theater, ecc (paradigma ***anytime anywhere***).
- Semantic Web e Web Services
  - La possibilità di usare le informazioni ed i servizi offerti dal Web non da parte di esseri umani direttamente, ma da parte di applicazioni che intermediano tra i dati e gli esseri umani, fornendo servizi a valore aggiunto con le informazioni altrui.

Che cos'è una pagina web

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><img border="0" src="<URL>"
width="288" height="263"></p>
<p> </p>
</body>
</html>
```

Web e HTTP

Terminologia essenziale

- Una pagina Web è costituita da oggetti
  - HTML file, JPEG image, Java applet, audio file,…
- Una pagina Web consiste di un file base HTML che fa riferimento ad altri oggetti
- Ogni oggetto è raggiungibile attraverso un Uniform Resource Locator (URL) (più in generale URI)
- Gli oggetti vengono richiesti dal browser web utilizzando un protocollo specifico

URL (Uniform Resource Locator)

Ogni URL è composto di tre parti:

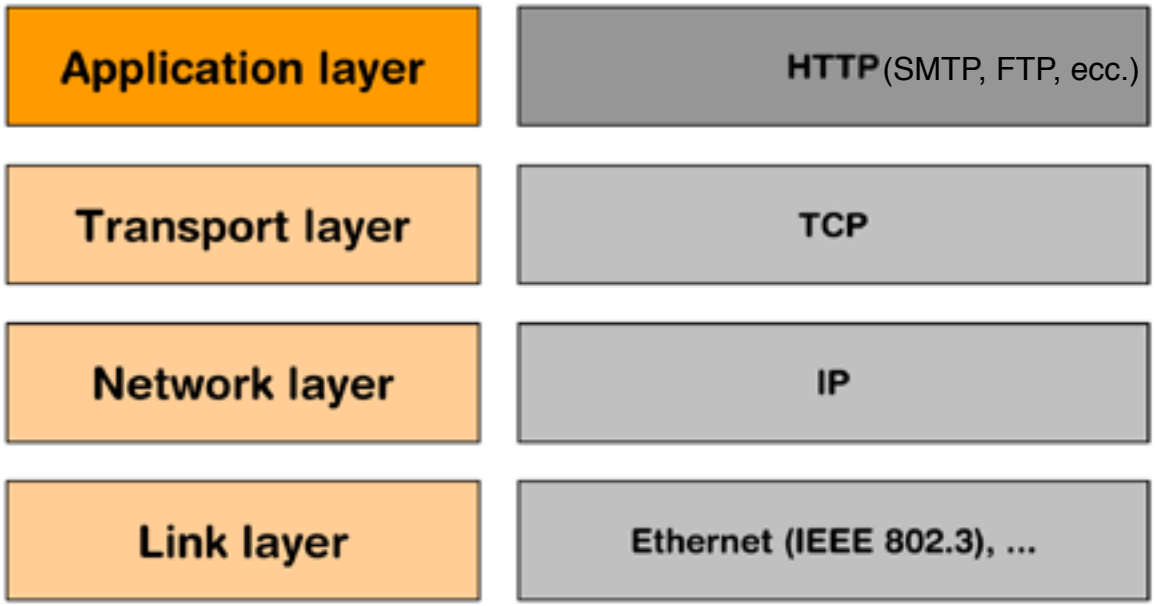
<protocollo>://<host name>/<pathname>

Esempio:

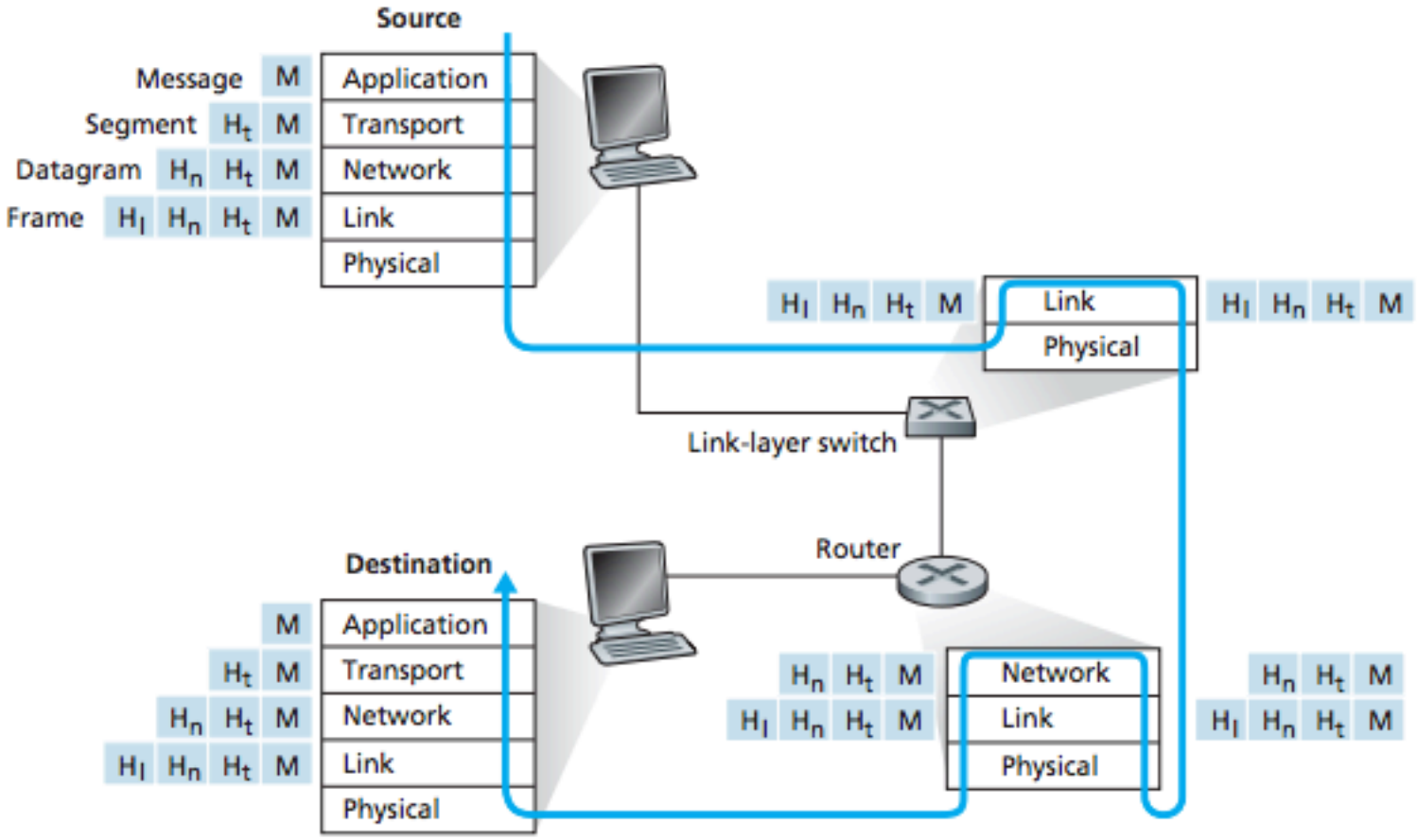
http://www.cnr.it/IngSoft/index.htm

domain name

Stack protocollare



Protocol layers

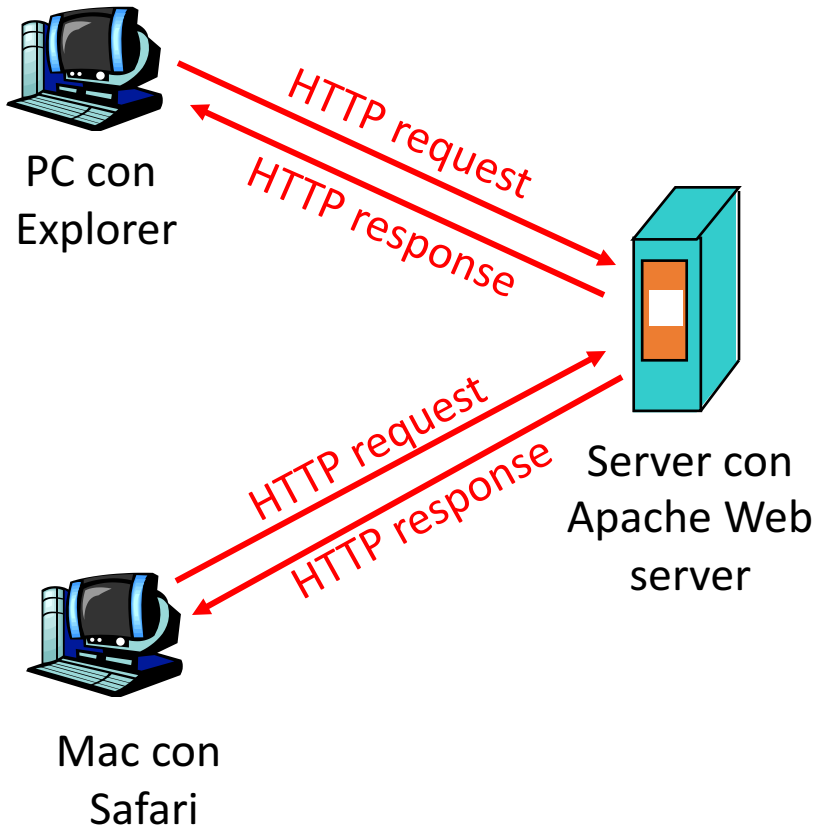


Terminologia

- Numero/indirizzo IP: è un'etichetta numerica che identifica univocamente un dispositivo detto host collegato a una rete informatica che utilizza l'Internet Protocol come protocollo di rete.
- Porta: Ogni processo locale che comunica con uno remoto viene identificato in una connessione TCP/IP tramite una porta.
  - **Well Known Ports**
  - 7 ECHO - Servizio Echo;
  - 20 FTP DATA - File Transfer Protocol Dati;
  - 21 FTP - File Transfer Protocol Controllo; 22 SSH - Secure Shell Remote Login Protocol
  - 23 TELNET - Telnet Protocol;
  - ecc
- Socket: La combinazione tra indirizzo IP, protocollo di trasporto e numero di porta prende il nome di Socket.

Protocollo HTTP: generalità

- Protocollo di livello applicativo per il Web
- modello client/server
  - *client:* Browser (user agent) che richiede, riceve e "mostra" oggetti Web
  - *server:* Web server che invia oggetti in risposta alle richieste
- HTTP 1.0: RFC 1945
- HTTP 1.1: RFC 2616 (obsolete 2068)
  - (compatibile con HTTP 1.0)



- Come fa il client a capire a quale server riferirsi e interrogare?
- Su una macchina host quanti web server è possibile fare operare contemporaneamente?
- Il server ricorda i client che lo hanno interrogato?

Protocollo HTTP: generalità

Usa il TCP:

- Il client inizia una connessione TCP (utilizzando un socket) verso il server sulla porta 80
- Il server accetta la connesione TCP dal client
- Vengono scambiati messaggi http (messaggi del protocollo di livello applicativo) tra il browser (client http) e il Web server (server http)
- La connessione TCP è chiusa

HTTP è "stateless"

- Il server non mantiene informazione sulle richieste precedenti del client

Nota

I protocolli che mantengono informazione di stato sono complessi (es. TCP) !

- ❑ la storia passata (stato) deve essere conservata

Connessioni HTTP

Connessione non persistente

- Per ogni connessione TCP viene inviato un singolo oggetto
- HTTP/1.0 usa una connessione non persistente

Connessione persistente

- Più oggetti possono essere inviati su una singola connessione TCP
- HTTP/1.1 usa connessioni persistenti come default

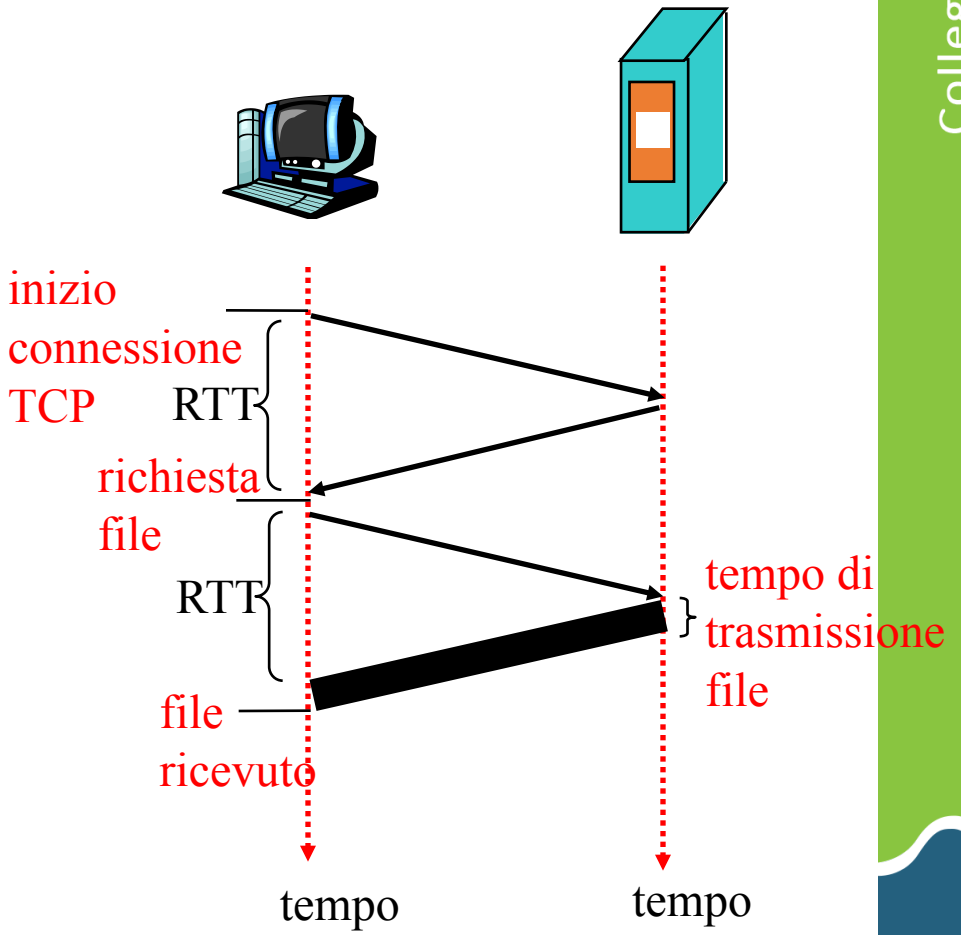
Tempo di risposta

Round Trip Time (RRT): tempo necessario a un piccolo pacchetto a viaggiare dal cliente al server e a tornare indietro

Tempo di risposta (Response time):

- un RTT per iniziare la connessione TCP
- un RTT per la richiesta HTTP e i primi byte di risposta
- tempo di trasmissione del file

totale = 2 RTT+ tempo di trasmissione



Connessioni persistenti e non persistenti

Connessioni non persistenti

- richiedono 2 RTT per oggetto
- Ogni oggetto subisce lo “slow start” TCP
- Comunque i browser spesso aprono connessioni TCP in parallelo per “scaricare” gli oggetti

Connessioni persistenti

- il server lascia la connessione aperta dopo aver inviato la risposta
- i messaggi successivi tra lo stesso client e server sono trasmessi su quella connessione

Persistenti senza pipelining

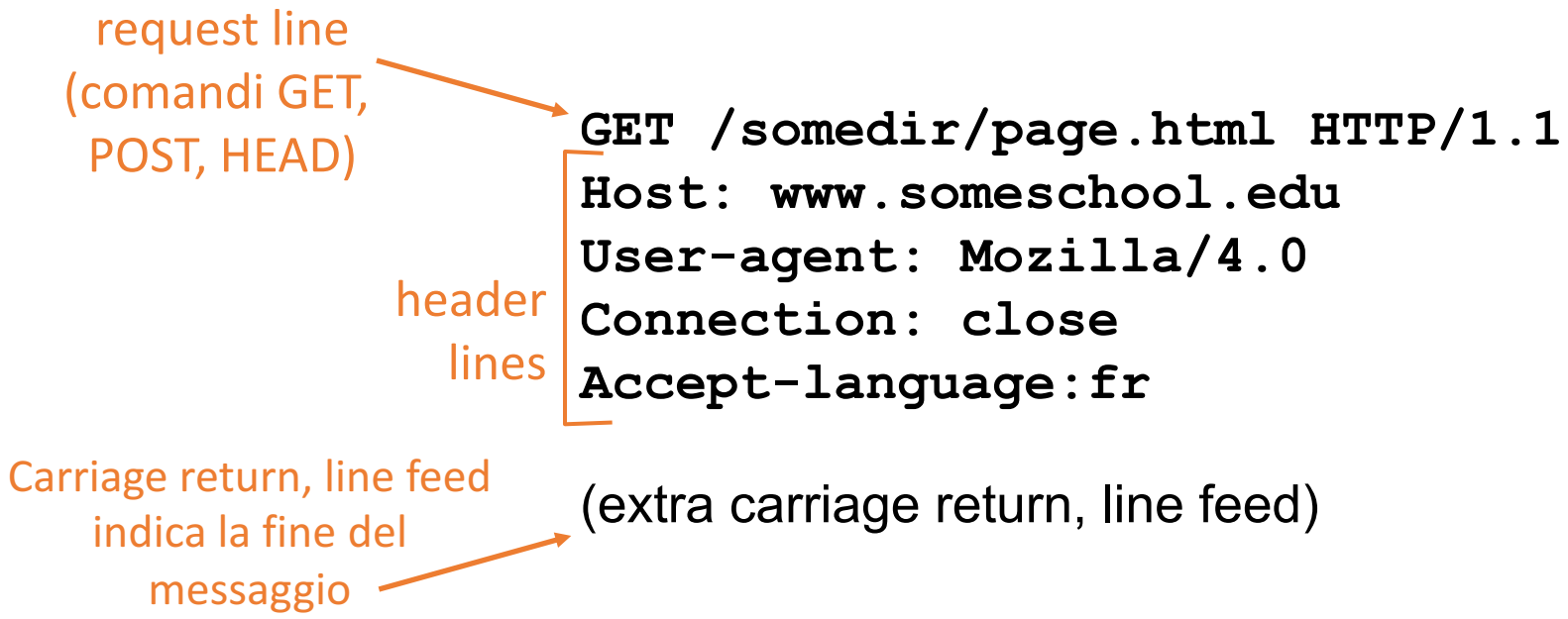
- il clienta invia una nuova richiesta solo quando la risposta precedente è arrivata
- 1 RTT per ogni oggetto referenziato

Persistenti con pipelining

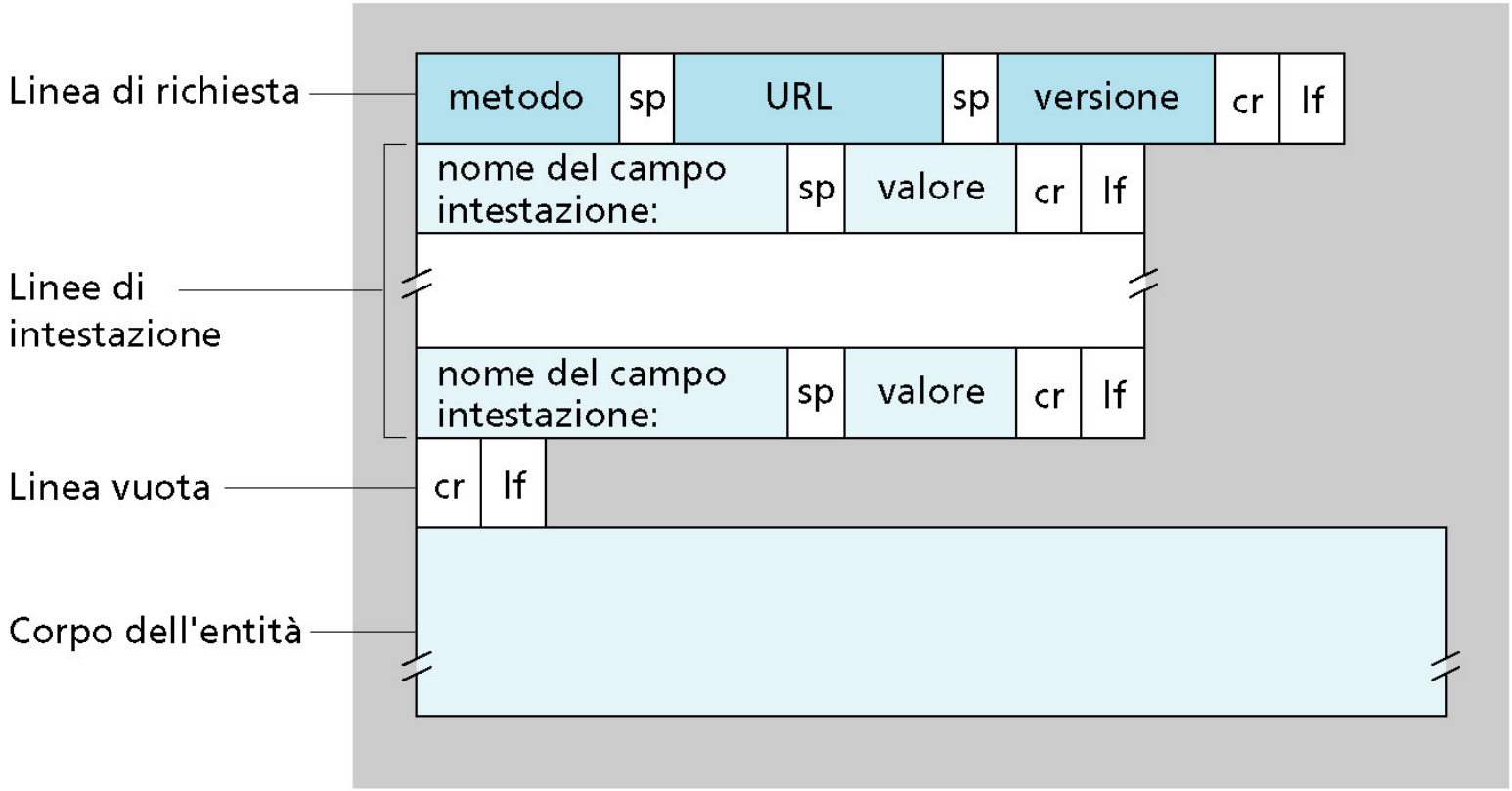
- default in HTTP/1.1
- il cliente invia una richiesta appena incontra il riferimento ad un oggetto
- richieste consecutive (back-to-back)

HTTP request message

- due tipi di messaggi HTTP: *request, response*
- HTTP request message:
  - ASCII (human-readable format)



HTTP request message: formato generale



Invio dei form

Metodo Post:

- Usato quando l'utente compila una form
- Il contenuto dei campi dei form sono inseriti nell'Entitiy Body
- Il comando richiede una pagina Web il cui contenuto dipende dalle informazioni nel campo body
- Es. query inviata ad un motore di ricerca

Metodo URL:

- Usa il metodo GET
- L'input è inserito nel campo URL della linea di richiesta:

```
www.somesite.com/animalsearch?monkeys&banana
```

Tipi di richiesta

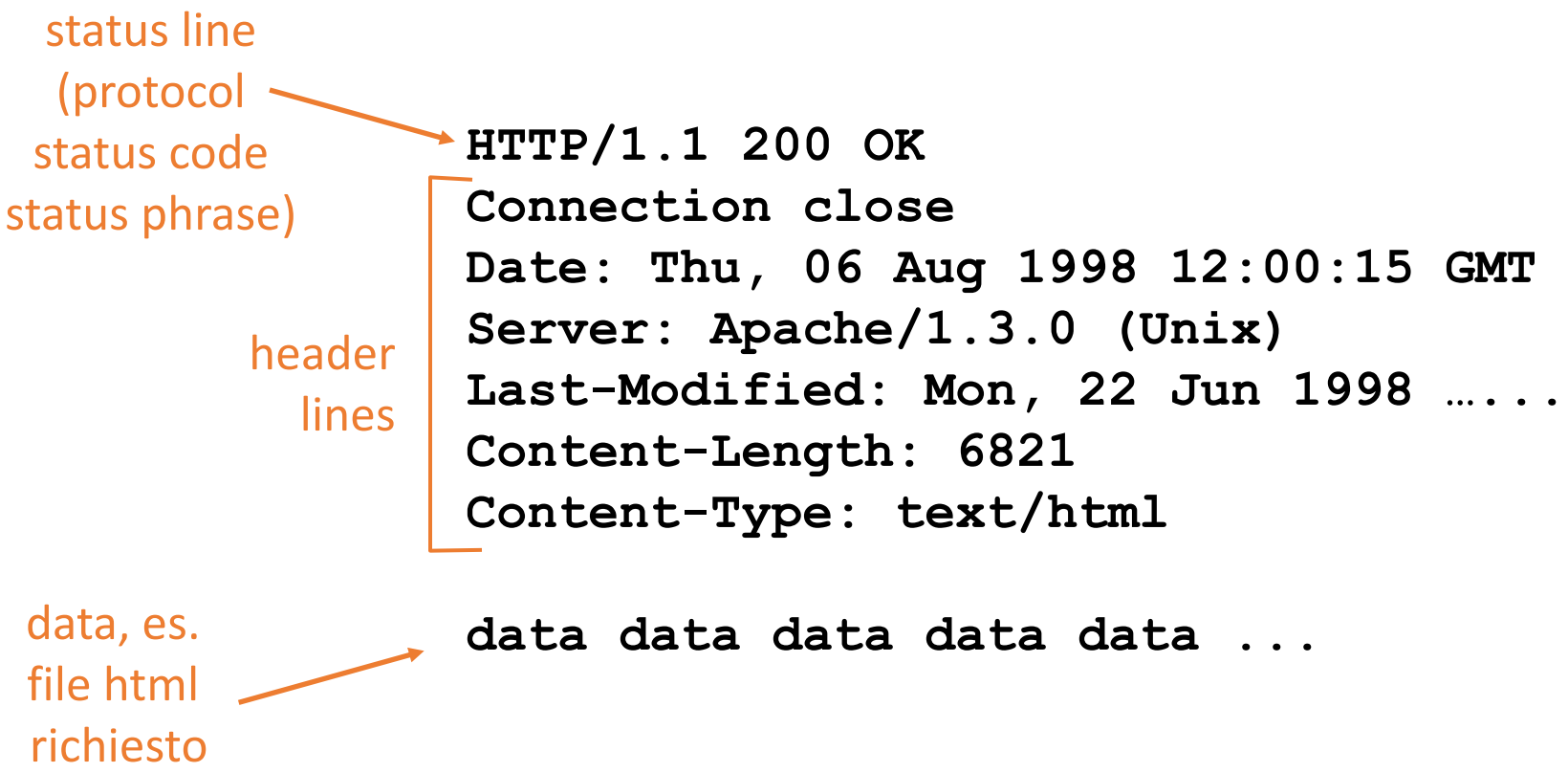
HTTP/1.0

- GET
- POST
- HEAD
  - chiede al server di rispondere solo con I metadati dell'header (tralasciando il corpo del messaggio)
  - usato spesso per debugging

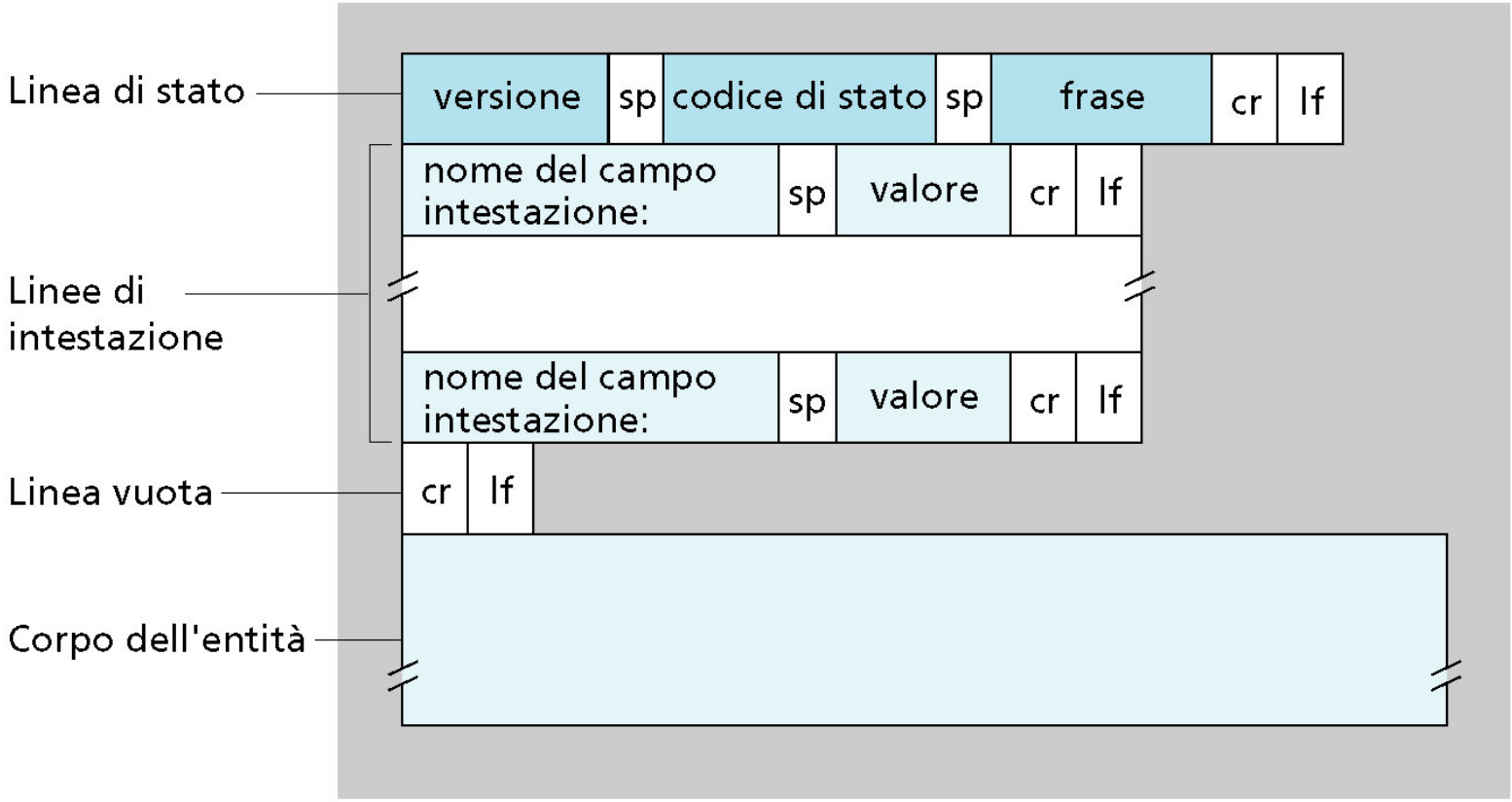
HTTP/1.1

- GET, POST, HEAD
- PUT
  - carica il file contenuto nel body nel path specificato dal campo URL
- DELETE
  - cancella il file specificato nel campo URL

HTTP response message



HTTP response message: formato generale



Codice di stato della risposta

- E’ un numero di tre cifre, di cui la prima indica la classe della risposta e le altre due la risposta specifica
- Esistono le seguenti classi:
- **1xx: Informational** una risposta temporanea alla richiesta, durante il suo svolgimento
- **2xx: Successful** il server ha ricevuto, capito e accettato la richiesta
- **3xx: Redirection** il server ha ricevuto e capito la richiesta, ma possono essere necessarie altre azioni da parte del client per portare atermine la richiesta
- **4xx: Client error** la richiesta del client non può essere soddisfatta per un errore da parte del client (errore sintattico o richiesta non autorizzata) −
- **5xx: Server error** la richiesta può anche essere corretta, ma il server non è in grado di soddisfare la richiesta per un problema interno (suo o di applicazioni invocate per generare dinamicamente risorse)

Esempi di codici di stato + frase

Prima riga del messaggio di risposta server->client.
Alcuni esempi:

200 OK

- Successo, oggetto richiesto più avanti nel messaggio

301 Moved Permanently

- L'oggetto richiesto è stato spostato. Il nuovo indirizzo è specificato più avanti (Location:)

400 Bad Request

- Richiesta incomprensibile per il server

404 Not Found

- Il documento non è stato trovato sul server

500 Internal Server Error

- Errore imprevisto che impedisce il servizio richiesto

505 HTTP Version Not Supported

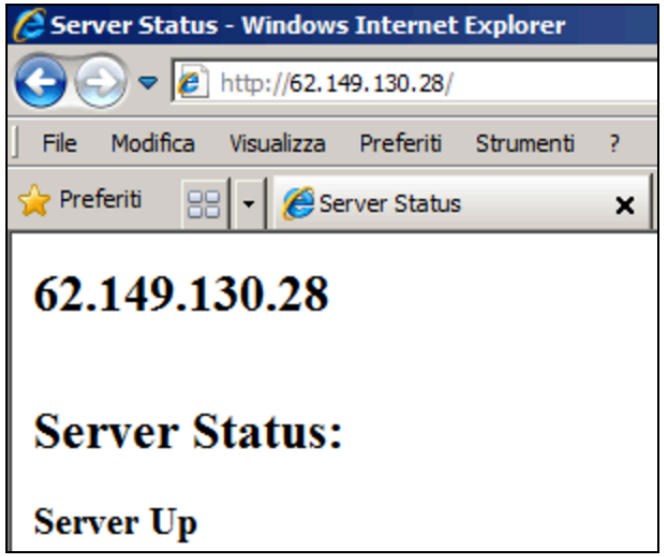
- versione richiesta dal protocollo HTTP non supportata dal server

Test HTTP mediante TELNET

- Usiamo per il nostro esempio il sito www.brescianet.com.
- Prima di tutto identifichiamo l'indirizzo IP del sito www.brescianet.com.
- In una finestra DOS digitiamo il comando ping

```
E:\Users\Administrator>ping www.brescianet.com

Esecuzione di Ping www.brescianet.com [62.149.130.28] con 32 byte di dati:
Risposta da 62.149.130.28: byte=32 durata=89ms TTL=119
Risposta da 62.149.130.28: byte=32 durata=47ms TTL=119
Risposta da 62.149.130.28: byte=32 durata=52ms TTL=119
Risposta da 62.149.130.28: byte=32 durata=50ms TTL=119

Statistiche Ping per 62.149.130.28:
Pacchetti: Trasmessi = 4, Ricevuti = 4,
Persi = 0 (0% persi),
Tempo approssimativo percorsi andata/ritorno in millisecondi:
Minimo = 47ms, Massimo = 89ms, Medio = 59ms
```

- L'IP che cerchiamo è quindi: **62.149.130.28**.
- Apriamo adesso **Internet Explorer** e digitiamo sulla barra degli indirizzi **http://62.149.130.28** .
- Il browser dovrebbe visualizzare la pagina in figura indicante lo status del server "multisito" [un solo IP n domini] che ospita "www.brescianet.com"

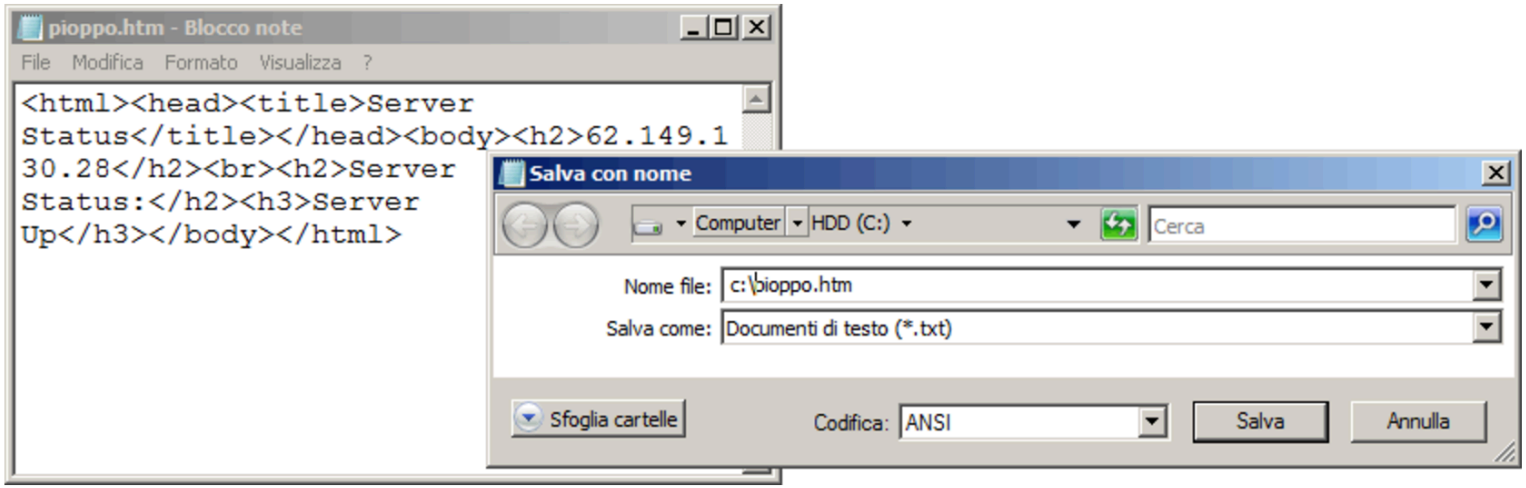


- Apriamo adesso una finestra DOS e digitiamo: Telnet 62.149.130.28 80 La finestra a questo punto si pulisce.
- Digitiamo quindi (rispettando il maiuscolo e il minuscolo): GET / HTTP/1.1 e battiamo invio (a video non si vede il testo che abbiamo battuto!).
- Scriviamo poi: Host: 62.149.130.28

```
E:\Users\Administrator>telnet 62.149.130.28 80

... il monitor su pulisce
... il testo in giallo non viene visualizzato ...

GET / HTTP/1.1<invio>
Host: 62.149.130.28<invio>
<invio>

HTTP/1.1 200 OK
Cache-Control: public
Date: Sun, 31 Jan 2010 17:35:24 GMT
Content-Length: 134
Content-Type: text/html; charset=utf-8
Expires: Sun, 31 Jan 2010 17:35:25 GMT
Last-Modified: Sun, 31 Jan 2010 17:35:24 GMT
Server: Microsoft-IIS/6.0
X-AspNet-Version: 2.0.50727
MicrosoftOfficeWebServer: 5.0_Pub
X-Powered-By: ASP.NET

<html><head><title>Server Status</title></head><body><h2>62.149.130.28</h2><br><h2>Server
Status:</h2><h3>Server Up</h3></body></html>
```

HTTP Response Headers

- La sequenza a video rappresenta la risposta [**HTTP Response Headers**] del WEB server (*IIS della microsoft versione 6.0 nel nostro esempio*) che eroga le pagine di **www.brescianet.com**. Vediamo di analizzare la parte di risposta contenente i tag HTML. Copiamo quindi nel blocco note la sequenza evidenziata in giallo e salviamola nel file response.htm.



- Vediamo un esempio analogo di HTTP Request mediante Telnet.
- Rispetto al precedente differisce solo per aver sostituito, nell'header tag Host, l'indirizzo IP reale con il nome effettivo del sito. La pagina HTML ricevuta questa volta corrisponde alla vera homepage del sito www.brescianet.com.
- Questo esempio evidenzia come il provider di brescianet.com (Aruba) riesce a gestire con un unico IP più siti sulla stessa macchina: sfrutta l'header Host per capire la pagina da inviare al client.

```
E:\Users\Administrator>telnet www.brescianet.com 80

.. il monitor su pulisce
... il testo in giallo non viene visualizzato ...

GET / HTTP/1.1<invio>
Host: www.brescianet.com<invio>
<invio>


HTTP/1.1 200 OK
Date: Sun, 31 Jan 2010 21:33:23 GMT
Content-Length: 1365
Content-Type: text/html
Content-Location: http://www.brescianet.com/index.htm
Last-Modified: Tue, 11 Aug 2009 09:13:31 GMT
Accept-Ranges: bytes
ETag: "601eb4fe631aca1:8a506"
Server: Microsoft-IIS/6.0
X-Powered-By: ASP.NET
MicrosoftOfficeWebServer: 5.0_Pub


<html>
<head>
<meta name="verify-v1" content="VVvE/ONgxck8aoKG/k1JkqCqxORatyd
<title>Appunti di INFORMATICA - Prof. Marco SECHI</title>
</head>

<FRAMESET rows='17,19,*' frameborder="0" framespacing="0" BORDE
<FRAMESET cols='*,120' BORDER=OFF>
<FRAME SRC=TITOLO.HTM NAME=Titolo marginwidth=0 marginheight=0
<FRAME SRC=webappl/CONTA.ASP NAME=Conta marginwidth=0 marginhei
</FRAMESET>
<frame name="bottoni" scrolling="no" noresize src="menuh.htm" m
>
```

Altro esempio HTTP Request

1. Telnet ad un Web server qualsiasi:

```
telnet www.iasi.cnr.it 80
```

Apre una connessione TCP al port 80
a www.iasi.cnr.it
Ogni carattere digitato è inviato
alla porta 80 si www.iasi.cnr.it

2. Digita una GET HTTP request:

```
GET /index3.html HTTP/1.0
```

Digitando questo messaggio
(carriage return due volte),
si invia questa GET request
minima (ma completa) al server HTTP

3. Si osservi la risposta del server

HTTP 1.0 definito in RFC 1945
HTTP 1.1 definito in RFC 2068

Interazione user-server: autorizzazione

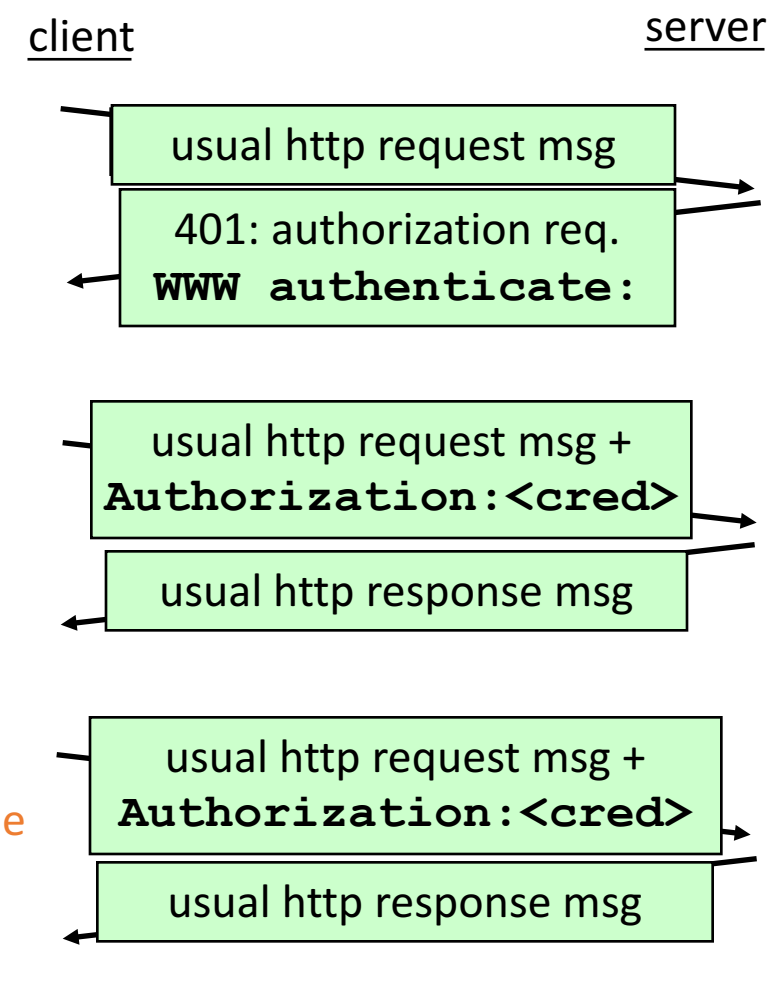
Autorizzazione: controllo dell'accesso agli oggetti del server

- authorization credentials: solitamente nome e password
- stateless: il cliente deve fornire l'autorizzazione per *ogni* richiesta
  - authorization: header line in ogni richiesta
  - se non c'è authorization: header il server rifiuta l'accesso e invia **`WWW authenticate:`**

    header nella risposta

Il browser memorizza nome & password in modo che l'utente non debba digitarli ogni volta

Esempio su: https://httpbin.org/



Interazione user-server: Cookies: mantenere lo “stato”

Molti dei più importanti siti web usano i cookie

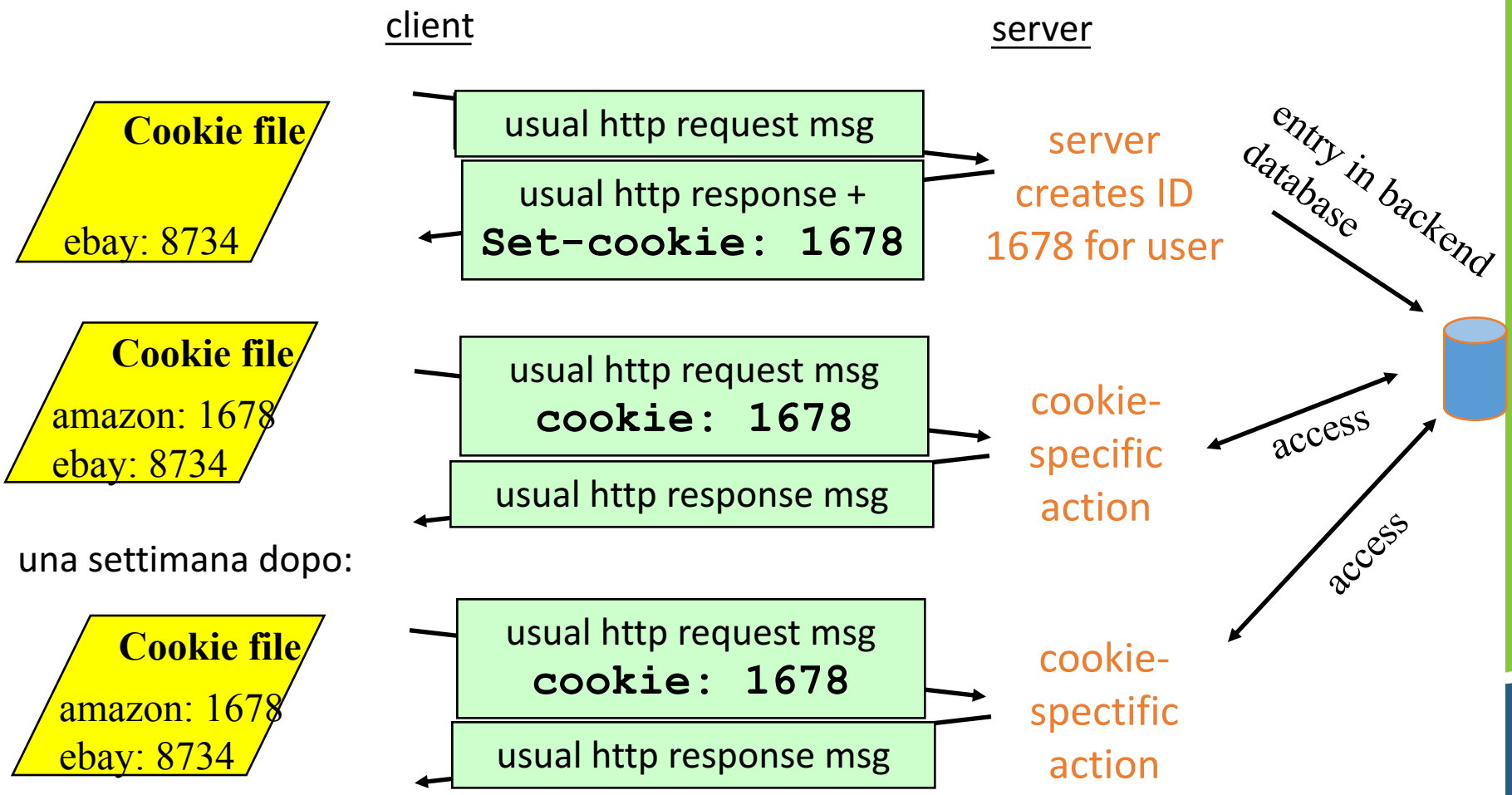
Quattro componenti:

1) cookie header line nel messaggio di **risposta** HTTP
2) cookie header line nel messaggio di **richiesta** HTTP
3) file cookie mantenuto nell’host e gestito dal browser
4) back-end database sul sito Web

Esempio:

- Susan accede Internet sempre dallo stesso PC
- Visita un sito di e-commerce per la prima volta
- Quando la richiesta iniziale HTTP arriva al sito, viene creata un’unica ID e una entry nel backend database per quella ID

Cookies: mantenere lo “stato” (cont.)



Cookies (cont.)

Utilizzo dei cookie:

- autorizzazione
- shopping carts
- one click shopping
- raccomandazioni
- stato sessione utente (Web e-mail)

Nota

Cookies e privacy:

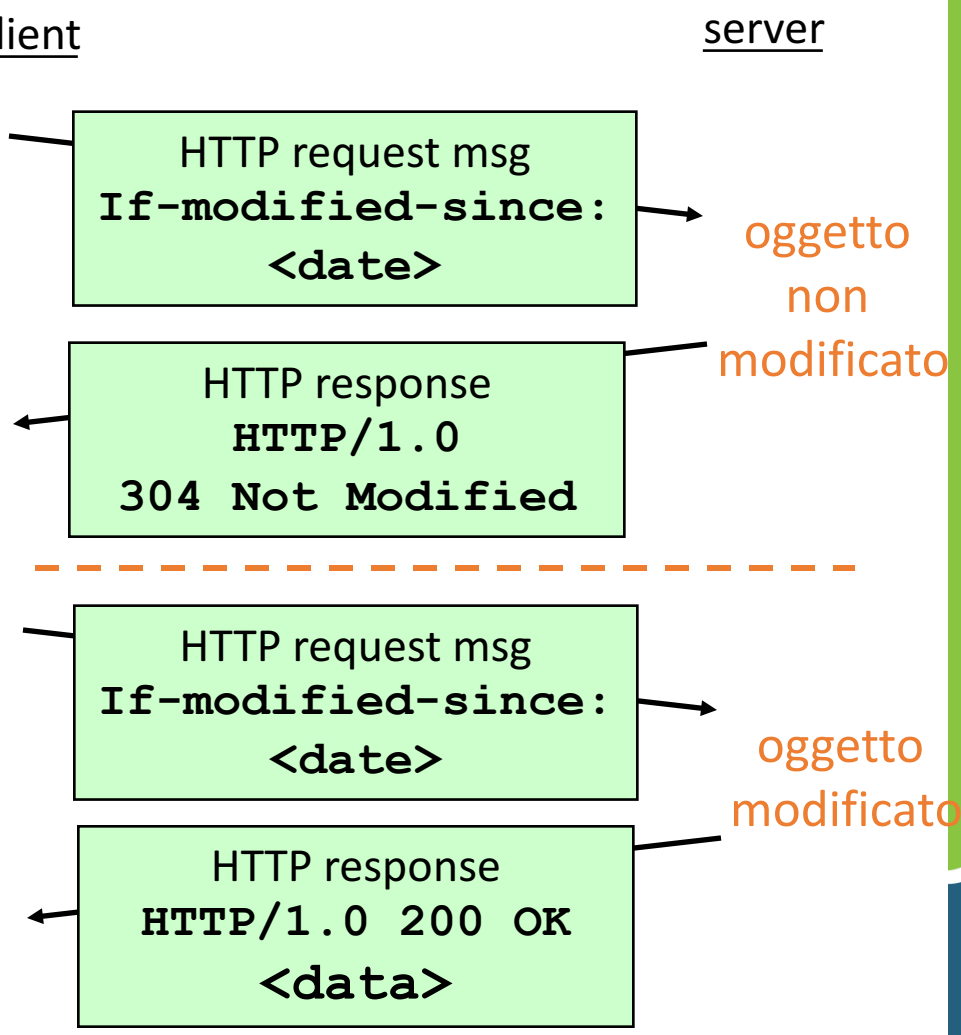
- i cookie permettono ai siti di imparare molte cose sull'utente
- se si forniscono dati personali, il sito web associa tali dati al cookie nel DB di back-end

GET condizionato: cache lato client

- Obiettivo: non mandare l'oggetto se il cliente ha una versione aggiornata in cache
- client: specifica la data della copia in cache nella richiesta HTTP

  `If-modified-since: <date>`
- server: la risposta non contiene l'oggetto se la copia in cache è aggiornata

  `HTTP/1.0 304 Not Modified`



Server Web

- Applicazione software
  - che fornisce accesso ad un sito Web
- Caratteristiche
  - fornisce servizi basati su HTTP
  - consente l'accesso a documenti statici
  - consente l'accesso a servizi interattivi (applicazioni) >> server applicativo

Server Web (1)

- Apache HTTP Server (httpd.apache.org)
  - open source
  - server HTTP
  - vari protocolli di interfaccia con server applicativi (es: CGI, PHP)
- Microsoft Internet Information Services
  - server HTTP
  - integrato con il server applicativo .NET

Server Web (2)

- Apache Tomcat (jakarta.apache.org)
  - server applicativo open source per Servlet e JSP
  - include server HTTP
- IBM WebSphere, Bea WebLogic ed altri
  - server applicativi commerciali per J2EE
  - includono server HTTP
- JBoss (www.jboss.org)
  - server applicativo open source per J2EE
  - include server HTTP

Architettura di un Server Web

- Contiene vari moduli
- Server HTTP
  - implementa il protocollo HTTP
  - include vari altri servizi; es: caching, logging
- Gestore del file system
  - contenuti statici salvati come file
- Server applicativo
  - gestore di applicazioni e componenti

Server Web

- In sintesi: Principali servizi del server
  - servizio HTTP verso il client (include autenticazione e autorizzazione)
  - gestione delle risorse sul file system
  - gestione delle applicazioni
  - registrazione degli accessi (logging)
  - gestione dei meccanismi di caching

Browser Web

- Principali servizi
  - consente di specificare le richieste (URL)
  - implementa il protocollo HTTP
  - visualizza il contenuto delle risposte e consente la navigazione
  - cache locale
  - altri servizi (preferiti, stampa, salva, ecc.)
- Browser diversi, diverse compatibilità

Pattern MVC

- Un pattern è una soluzione “tipo” ad un certo tipo di problema ricorrente.
- Model – View – Controller
- Separare i dati e la business logic che opera su di essi dalla loro visualizzazione e dalla modalità di interazione con essi.

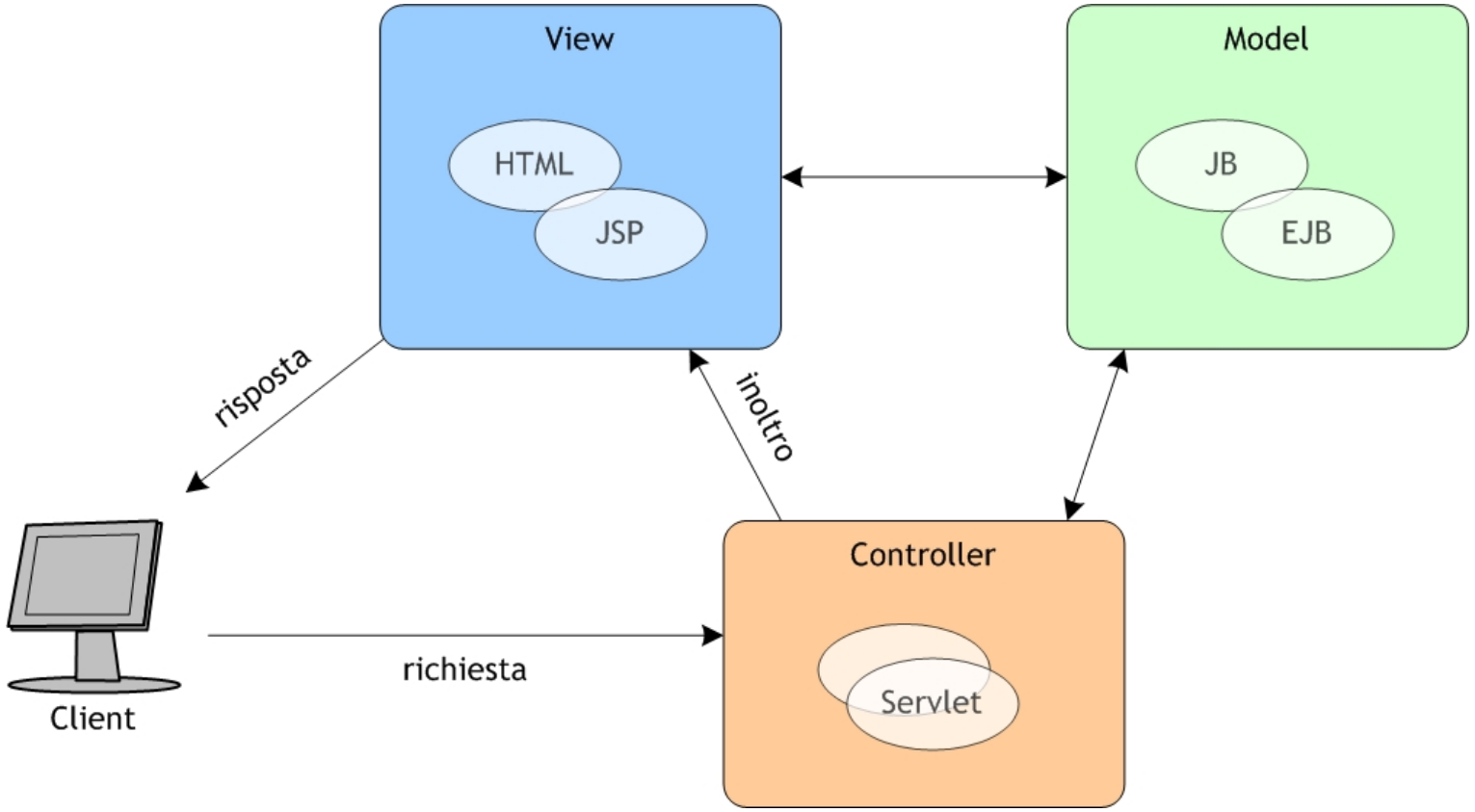
Pattern MVC (cont.)

- View
  - Il display del modello nell'interfaccia utente
  - Es. una pagina HTML che visualizza alcuni dati tratti dal modello, come per esempio i dati relativi a un libro in un online bookstore
- Controller
  - Gestisce l'interazione con l'utente Seleziona le parti del modello rilevanti ad un'operazione
  - Effettua i cambiamenti ai dati del modello relativi alle operazioni
  - Crea/modifica le view di risposta
- Model
  - Modello del sistema di cui ne rappresenta lo stato (tipicamente il database)

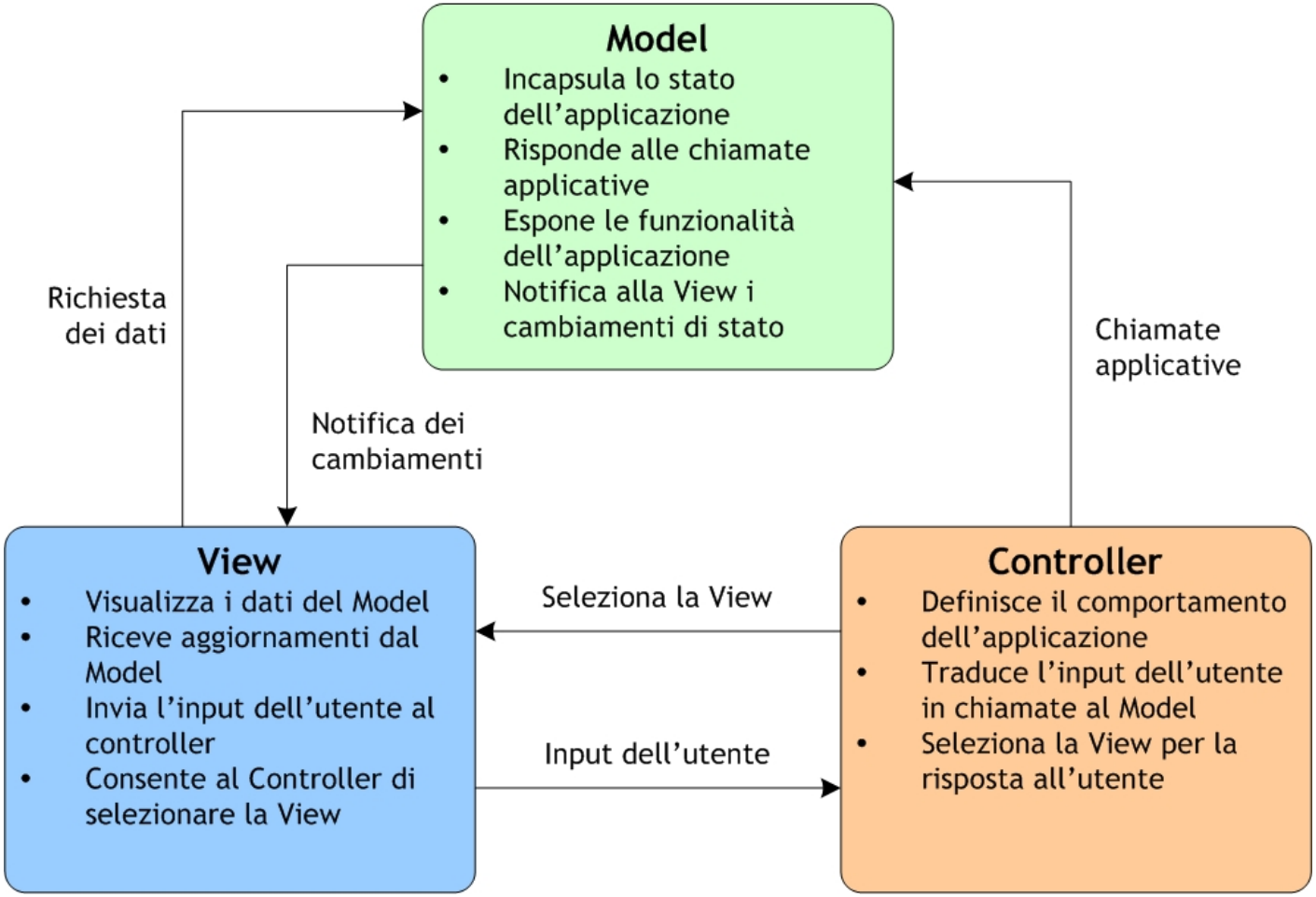
Pattern MVC (cont.)

- Quando usarlo
  - Tutte le volte che c'è bisogno di separare la presentazione dalla logica applicativa
  - La separazione fra View e Controller può essere ignorata in casi semplici

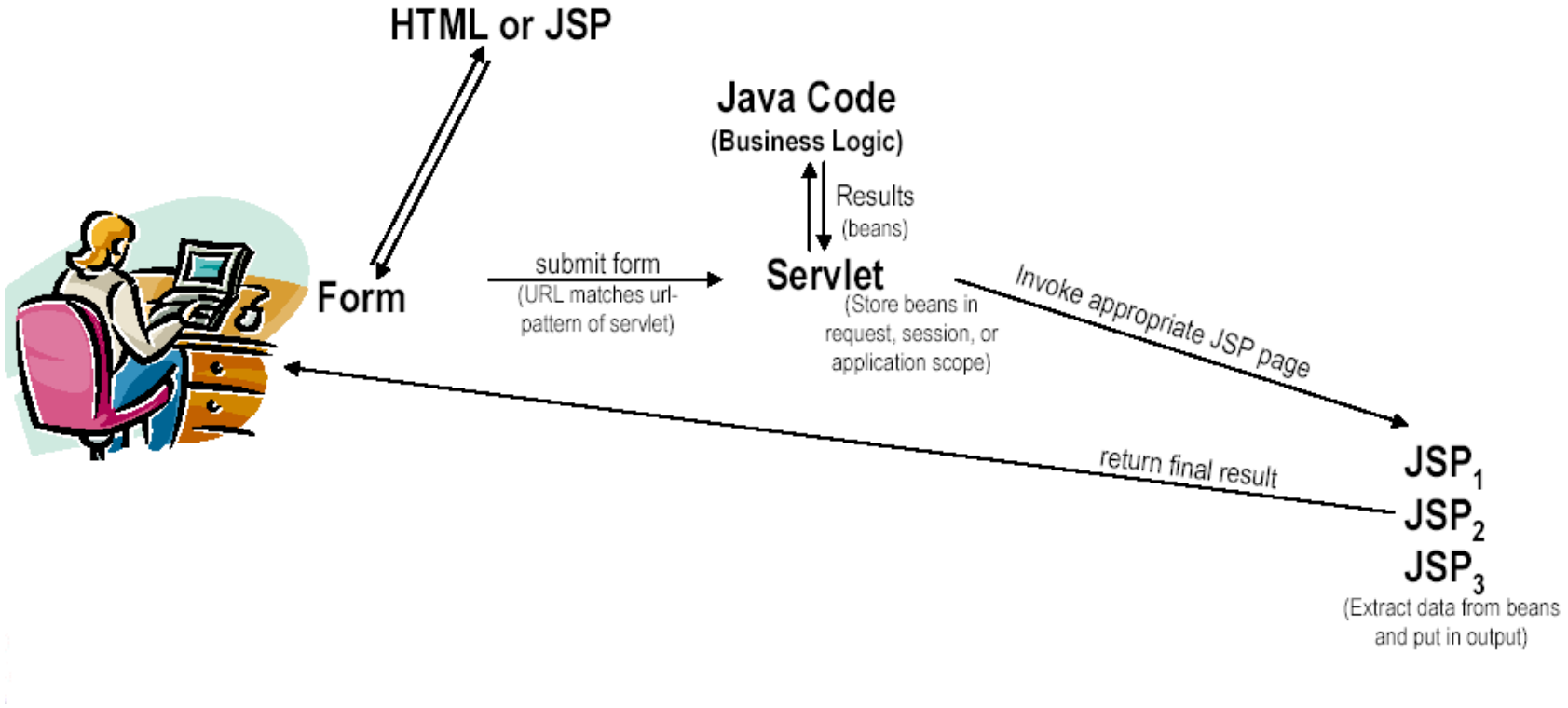
Pattern MVC (cont.)



Pattern MVC (cont.)



MVC Flow Control

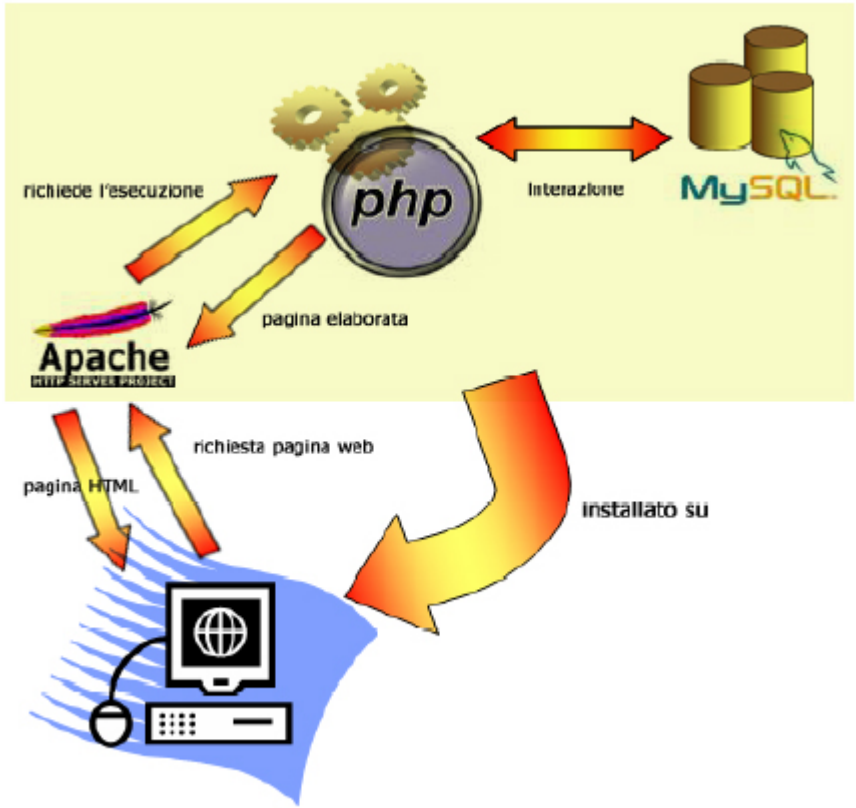


WAMP

- Prepariamo l’ambiente di sviluppo:
- WAMP è l'acronimo di Windows, **Apache**, MySQL, PHP
- Requisiti:
  - Windows XP Home o Professional

Installazione WAMP

- Installando Apache/2.4.2 si potrà disporre di un server web in locale
- con MySQL/MariaDB avremo anche un server di database. Con PHP 5.6.28 avremo a disposizione di un engine per la realizzazione e l'elaborazione di contenuti dinamici sul server web.
- Dagli script PHP sarà, previa un'opportuna configurazione, possibile connettersi al server MySQL ed interagire con la base dati.



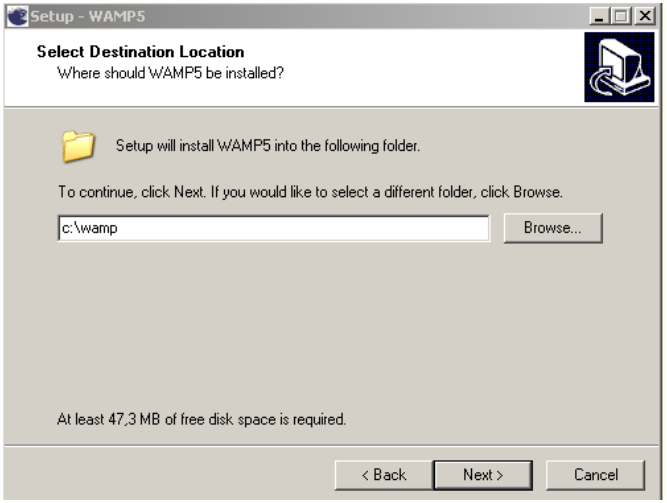
Installazione WAMP

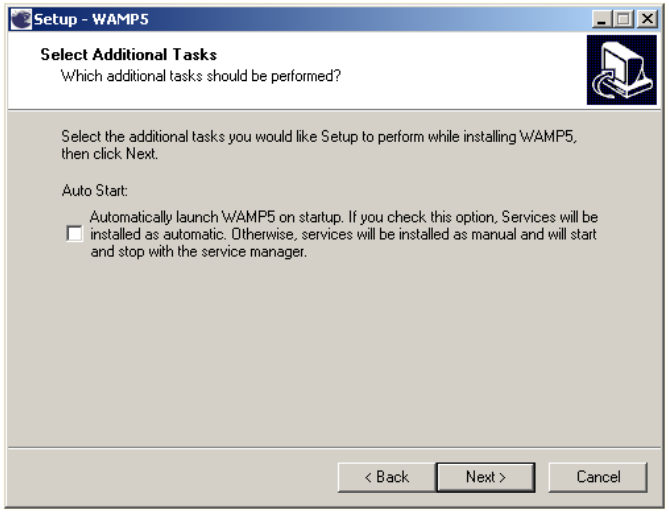
- Scaricare il paccketto **WAMP5 3.0.x** al seguente: http://www.wampserver.com/en/download.php
- WAMP5 configurerà automaticamente i seguenti software consentendogli di integagire tra di loro:

  Apache 2.x.x.
  PHP 5.x.x
  MySQL 5.x.x
  PHPmyadmin
  Adminer
  Wampserver service manager

Installazione WAMP

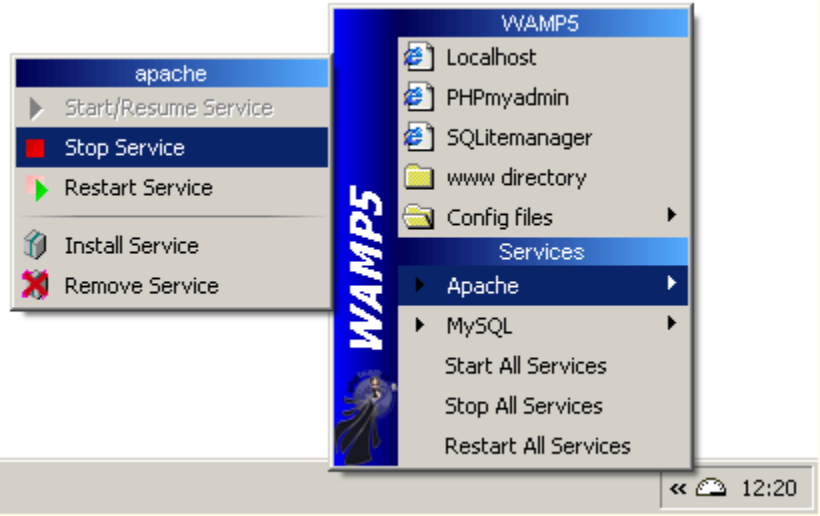
- Eseguire il setup di WAMP
- Selezionare la directory di installazione (va bene quella di default)
- Scegliere se avviare i servizi di WAMP in maniera automatica o manuale.





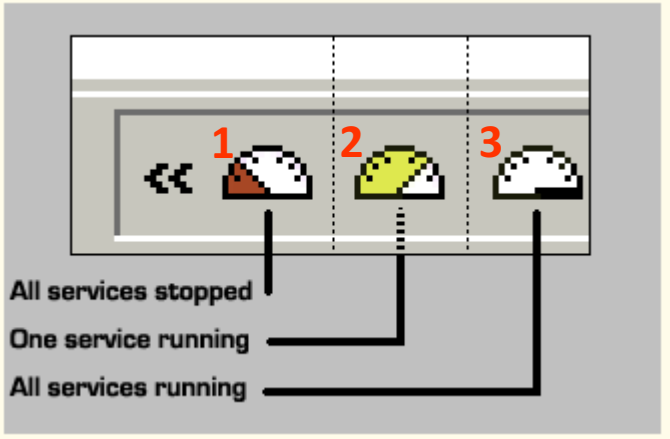
Utilizzo di WAMP

- Nel traybar appare un'icona a forma di orologio che indica lo stato di WAMP
- Cliccando sull'icona con il tasto dx possiamo interagire con i server (Apache, Mysql)
- Per ognuno possiamo:
  - Avviare il servizio
  - Stoppare il servizio
  - Riavviare il servizio



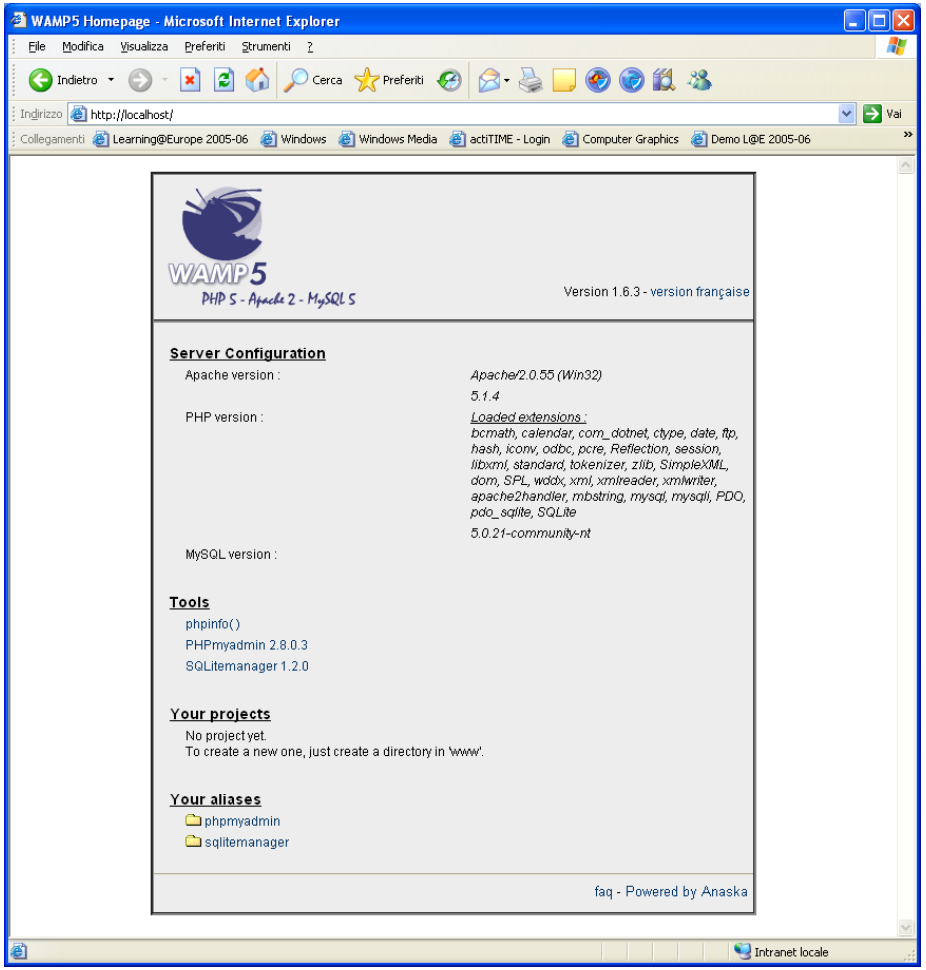
Stato di WAMP

- L'icona nella traybar indica:
  1. Che i servizi sono bloccati
  2. Che solo uno dei due servizi (o server) è in funzione
  3. Entrambi i servizi sono avviati



Verifica del funzionamento

- Per verificare che tutto sia installato correttamente digitare nel proprio browser l'indirizzo:
  - http:// localhost
- La pagina visualizzata (qui di fianco) dovrebbe mostrare le versione dei servizi installati



XAMPP Installation

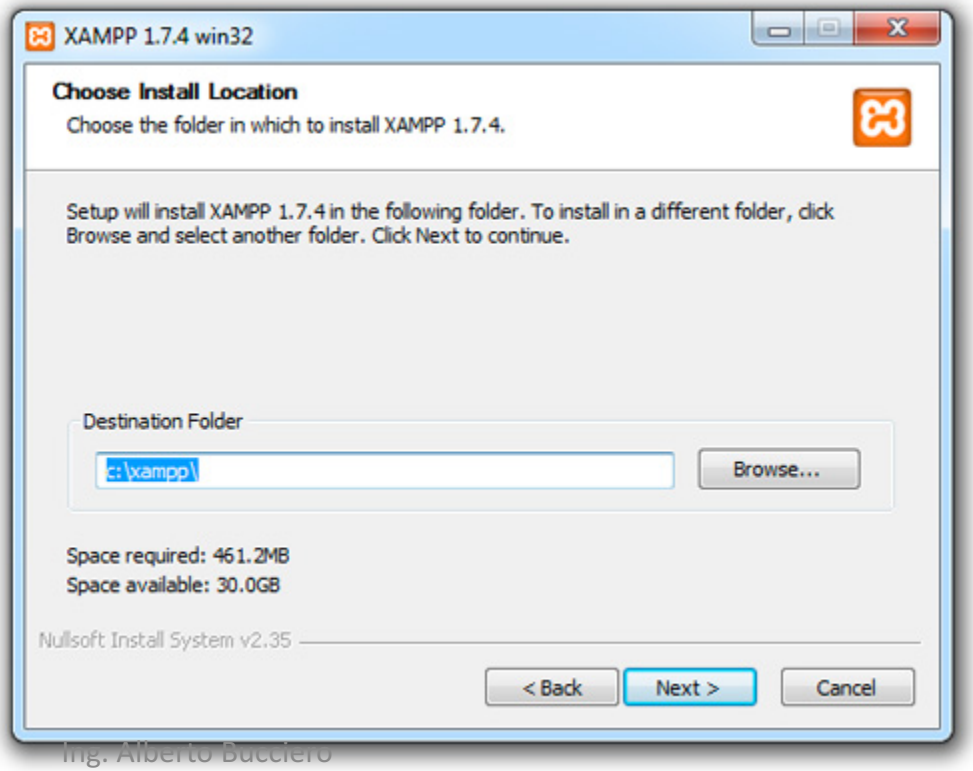
- **Installing with Windows :**

- Go to the XAMPP website, choose XAMPP Windows.
- Choose latest stable version and click the installer file to initiate download.
- XAMPP for Windows exists in three different flavors:
  - Installer : Probably the most comfortable way to install XAMPP.
  - ZIP : For purists: XAMPP as ordinary ZIP archive.
  - 7zip : For purists with low bandwidth: XAMPP as 7zip archive.
- You can also download XAMPP for Windows Add-Ons :
  - Tomcat Add-On
  - Perl Add-On

XAMPP Installation

- **Installation with the Installer :**

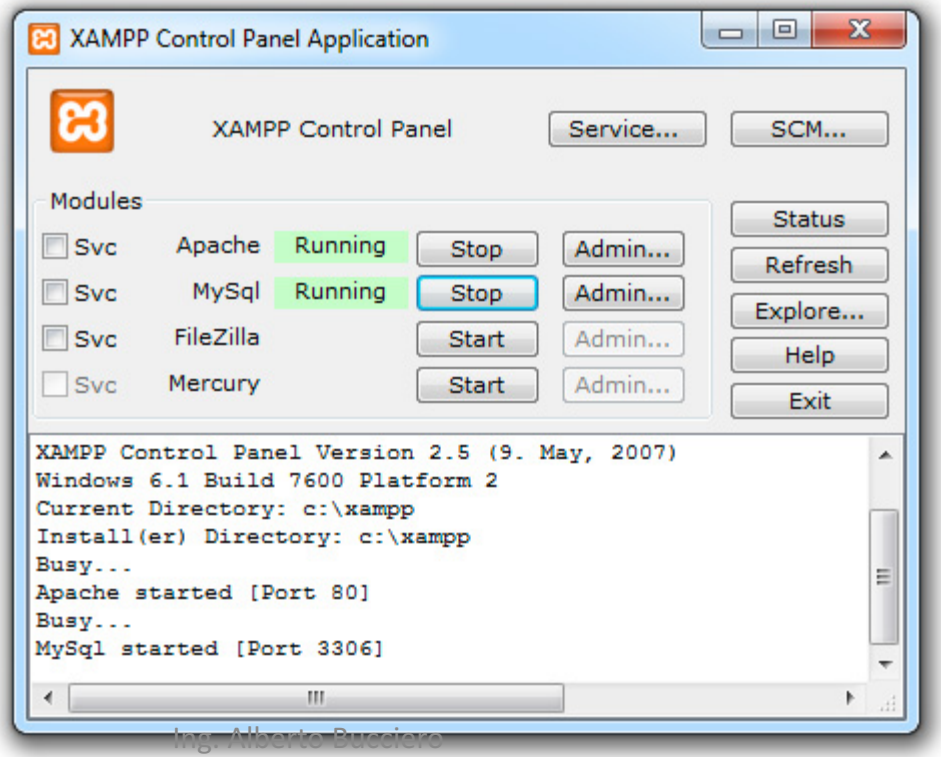
- Using the installer version is the easiest way to install XAMPP.



XAMPP Installation

- **Installation with the Installer :**

- After the installation is complete, you will find XAMPP under Start | Programs | XAMPP. You can use the XAMPP Control Panel to start/stop all server and also install/uninstall services.



XAMPP Installation

- **Installation with the Installer :**

- The XAMPP control panel for start/stop Apache, MySQL, FilaZilla & Mercury or install these server as services.

XAMPP Installation

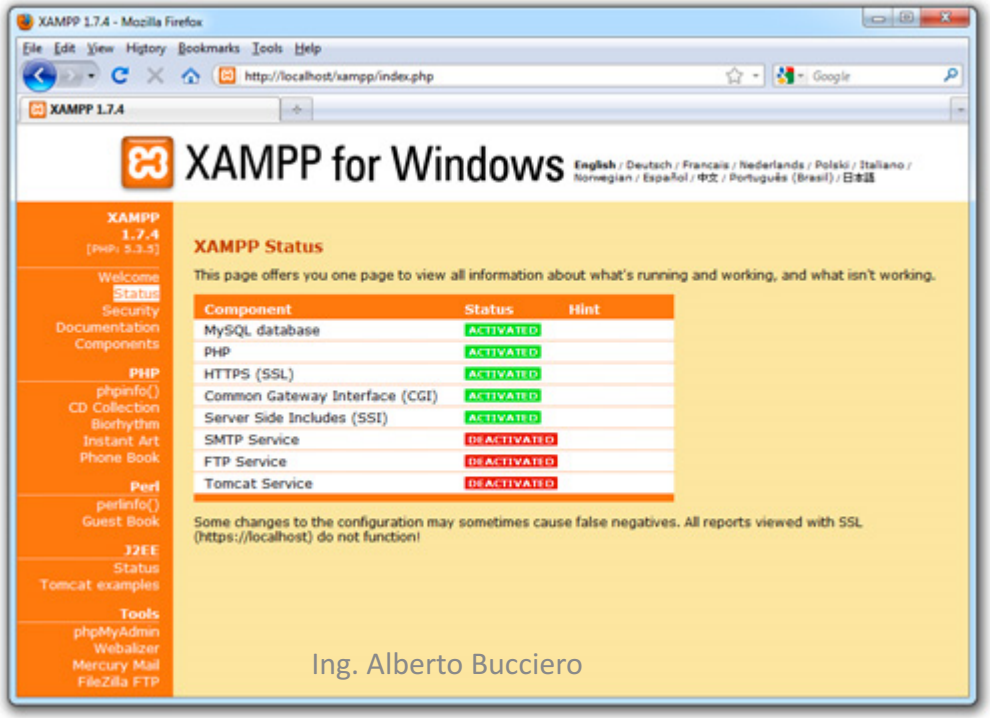
- **Installation without the Installer :**

- Unzip the zip archives into the folder of your choice. XAMPP is extracting to the subdirectory "[Ziel]\xampp" below the selected target directory. Now start the file *"setup_xampp.bat"*, to adjust the XAMPP configuration to your system.
- If you choose a root directory "C:\" as target, you must not start "setup_xampp.bat".
- If you extract XAMPP in a top level folder like "C:\" or "D:\", you can start most servers like Apache or MySQL directly without execution of the file "setup_xampp.bat".

XAMPP Configuration

- **Testing XAMPP:**

- After starting of Apache (and MySQL), go to the address http://localhost/ or http://127.0.0.1/ in your browser and examine all of the XAMPP examples and tools.



XAMPP Installation

- **Installing with Linux :**

- Go to the XAMPP website, choose XAMPP Linux.
- Choose latest stable version and click the installer file to initiate download.
- XAMPP for Linux exists in the form tar.gz
- You can also download Upgrade package from 1.7.3a version to 17.4 version.

XAMPP Installation

- **Installing with Linux :**

- After downloading simply type in the following commands:
- Go to a Linux shell and login as the system administrator root.
- Extract the downloaded archive file to /opt :

```
tar xvfz xampp-linux-1.7.4.tar.gz -C /opt
```

- XAMPP is now installed below the /opt/lampp directory.

XAMPP Configuration

- **Starting :**

- To start XAMPP simply call this command :

```
/opt/lampp/lampp start
```

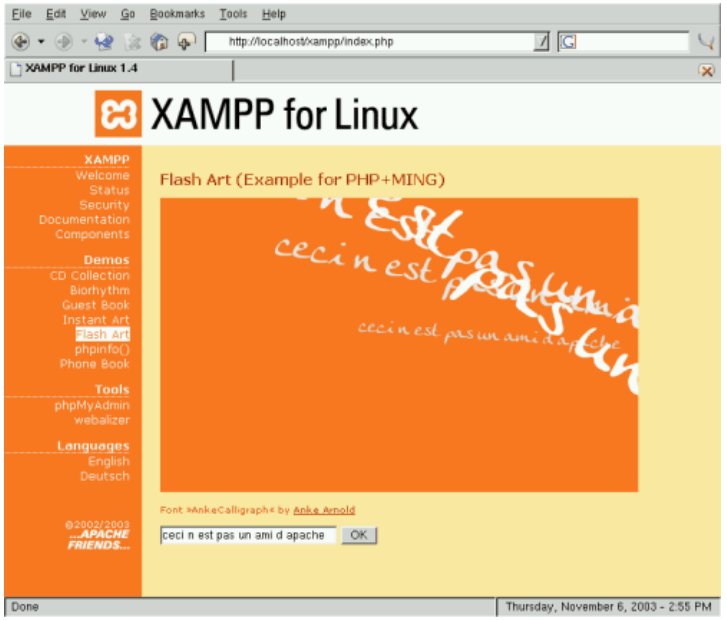
- You should now see something like this on your screen :

```
Starting XAMPP 1.7.4...
LAMPP: Starting Apache...
LAMPP: Starting MySQL...
LAMPP started.
```

- If you get any error messages please take a look at the Linux FAQ.

XAMPP Configuration

- **Testing :** `http://localhost`
- Type in the following URL at your favourite web browser :
- You should see the start page of XAMPP containing some links to check the status of the installed software and some small programming examples :



XAMPP Configuration

- **Stopping :**

- To stop XAMPP simply call this command :

```
/opt/lampp/lampp stop
```

- You should now see something like this on your screen :

```
Stopping LAMPP 1.7.4...
LAMPP: Stopping Apache...
LAMPP: Stopping MySQL...
LAMPP stopped.
```

- XAMPP for Linux is now stopped.

Riferimenti e bibliografia

- The web and HTTP: http://www.nylxs.com/docs/cmpnet.pdf
- Appunti di informatica - Prof Marco Sechi: http://www.brescianet.com/appunti/sistemi/http_protocol.htm