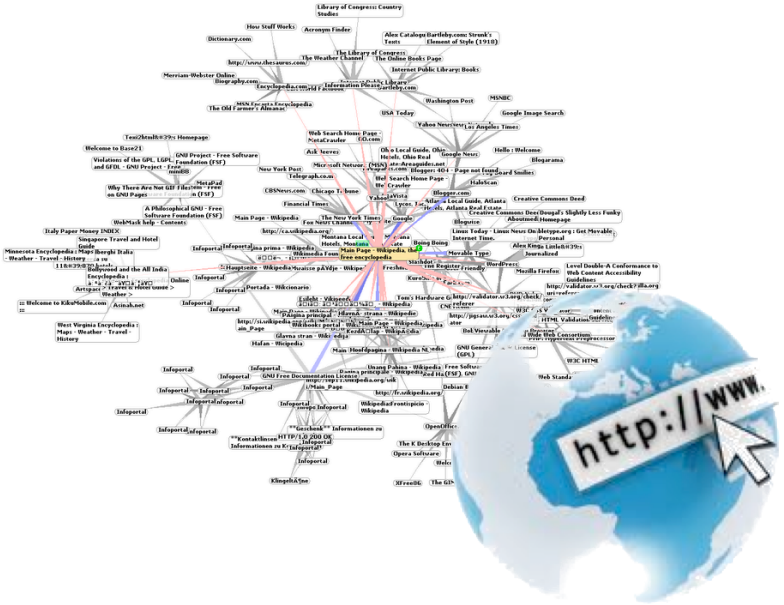


Reti di Calcolatori

HTTP e il World Wide Web

Il World Wide Web

- Nato nel 1989 al CERN di Ginevra come mezzo per scambiare informazioni (Tim Berners-Lee, Robert Cailliau)
  - Sistema di documenti ipertestuali e risorse distribuite sulla rete Internet collegate fra loro con hyperlink
  - Pensato come sistema di gestione dell'informazione che sfrutta le infrastrutture di rete per facilitare la pubblicazione, distribuzione, condivisione e accesso alle informazioni
  - La struttura a ragnatela (web) definita dagli hyperlink è descritta dal grafo del Web (webgraph)
    - i nodi del grafo sono le pagine (risorse) Web
    - gli archi diretti sono gli hyperlink presenti nelle pagine
  - Il World Wide Web Consortium (www.w3c.org) è l'organizzazione che sovrintende allo sviluppo degli standard per il Web

IL Web come applicazione

- Il Web è nato come servizio disponibile sulla rete Internet
  - E' reso disponibile da un insieme di applicazioni server (i server Web)
  - E' accessibile con un'applicazione client con interfaccia grafica (il browser Web)
  - Si basa sulla tecnologia degli ipertesti realizzata con un linguaggio pensato per questa applicazione (HTML - HyperText Markup Language)
    - Dalla prima proposta il linguaggio HTML si è evoluto per gestire la crescente complessità richiesta nella creazione dei documenti (attualmente è in uso la versione 5)
  - Definisce una modalità di indirizzamento delle risorse con l'Uniform Resource Locator (URL)
  - Utilizza un protocollo applicativo pensato per questa applicazione HyperText Transfer Protocol (HTTP)
    - E' diventato uno strumento diffuso per l'accesso a risorse su rete Internet

Evoluzione del Web

- Web 1.0
  - L'idea originaria del Web è come collezione di documenti statici
  - In teoria chiunque avrebbe potuto essere autore e pubblicare pagine sul Web
    - Gli strumenti per la creazione di pagine HTML si sono evoluti dagli editor di testo ad applicazioni WYSIWYG (WhatYouSeeIsWhatYouGet)
- Web 1.x
  - Le tecnologie del Web si sono evolute progressivamente per sfruttare le potenzialità di questo paradigma di accesso all'informazione
    - Possibilità di pubblicare informazioni in vari formati (immagini, video, audio,..)
    - Estensione dei protocolli per includere nuove funzionalità (HTTP1.1, supporto per i servizi interattivi con i Form HTML e l'architettura CGI)

Il Web "dinamico"

- Il Web assume una natura dinamica/interattiva
  - Generazione dinamica delle pagine Web Server-Side
    - Sviluppo di linguaggi di script HTML embedded (PHP, ASP)
    - Sviluppo di nuove tecnologie che trasformano i server Web da archivi di documenti ad application server (Servlet engines, JSP Java Server Pages)
  - Programmazione e scripting Client-Side in pagine Web interattive
    - Sviluppo della tecnologia Java e delle applet
    - Supporto per l'esecuzione di linguaggi di scripting da parte dei browser Web (javascript)
    - Possibilità di sviluppare "pagine Web" che in realtà sono applicazioni in esecuzione nel browser e che assumono sempre più l'aspetto di applicazioni native
  - Il Web si trasforma in una piattaforma applicativa per sviluppare servizi

Web computing

- Le tecnologie alla base del Web sono divenute il supporto per la diffusione di piattaforme di servizi distribuiti
  - I Web Services utilizzano gli standard del Web (HTTP, URI, XML)
- Gli obiettivi del Web computing sono
  - Fornire l'accesso a servizi software (applicazioni) resi disponibili su rete Internet
  - Dare la possibilità di combinare servizi esistenti (API di librerie di servizi) per costruire applicazioni complesse (mashup)
    - es. www.programmableweb.com
  - Eventualmente pagare il software in base all'uso
  - I servizi accessibili agli utenti tramite pagine Web sono resi disponibili anche come API
    - es. API di google maps (http://developers.google.com/maps/)

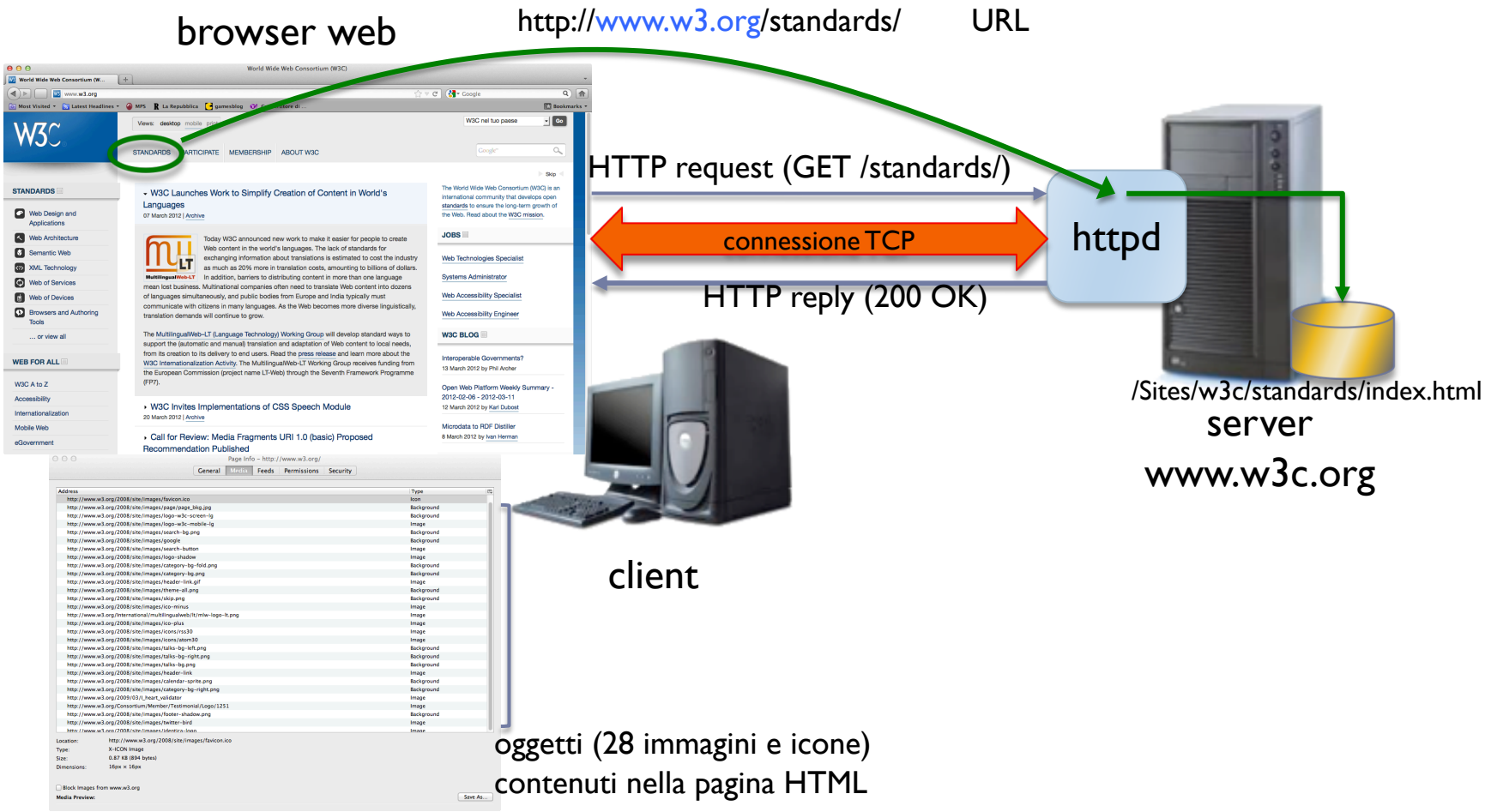
Web 2.0 e 2.x

- Il numero di versione indica una modifica nel modo di usare il Web e nei contenuti, metodologie e strumenti
  - Non esiste una versione del Web come quella dei programmi!
  - L'infrastruttura del Web è sempre la stessa (Internet e risorse collegate con hyperlink o riferite da URL)
  - Quello che cambia è l'esperienza che gli utenti hanno nel Web
  - Il Web è divenuto una piattaforma per lo sviluppo e la distribuzione di servizi e applicazioni
  - L'idea di Web 2.0 è legata ad ruolo centrale di ogni singolo utente
    - Sviluppo di reti sociali, tagging collettivo, forum, blog, wiki
  - Il Web 2.0 è più basato sui dati che sulle pagine (spostamento da HTML a XML e uso di javascript/Ajax lato client)
    - Es. Content Syndication - redistribuzione dei contenuti Web per mezzo di canali/feed (RSS - RDF Site Summary)

HTTP

- HyperText Transfer Protocol - HTTP - è il protocollo utilizzato per scambiare risorse sul Web
  - E' basato sul modello client/server (browser web/server web)
  - Utilizza TCP come protocollo di trasporto
    - Il server Web è in ascolto su una porta TCP (80 per default)
    - Il client apre una connessione TCP verso il server Web utilizzando le informazioni presenti negli hyperlink/URL
    - Il client invia la richiesta per la risorsa (la richiesta è un messaggio di testo ASCII)
    - Il server risponde inviando la risorsa richiesta o con un messaggio di errore
  - Non ci sono vincoli sul tipo di dati che possono essere trasferiti
    - testo, testo formattato HTML, dati binari (immagini, dati compressi, video,...)
    - in generale una pagina web può essere composta di più oggetti (il file base HTML, immagini, ecc.) che sono trasferiti con richieste distinte

Trasferimento HTTP



Uniform Resorce Locator (URL)

- Un URL (Uniform Resource Locator) individua univocamente una risorsa su web
  - Permette di descrivere in modo "uniforme" la posizione di una risorsa sul Web rispetto al procollo con cui si può accedervi
  - Contiene eplicitamente il riferimento all'host che la rende disponibile attraverso il suo nome DNS e opzionalmente alla porta
    - Non è un riferimento trasparente rispetto a spostamento/rimozione della risorsa (broken links)

```
protocol://host.domain:port/resource-path?query_string#fragment
```

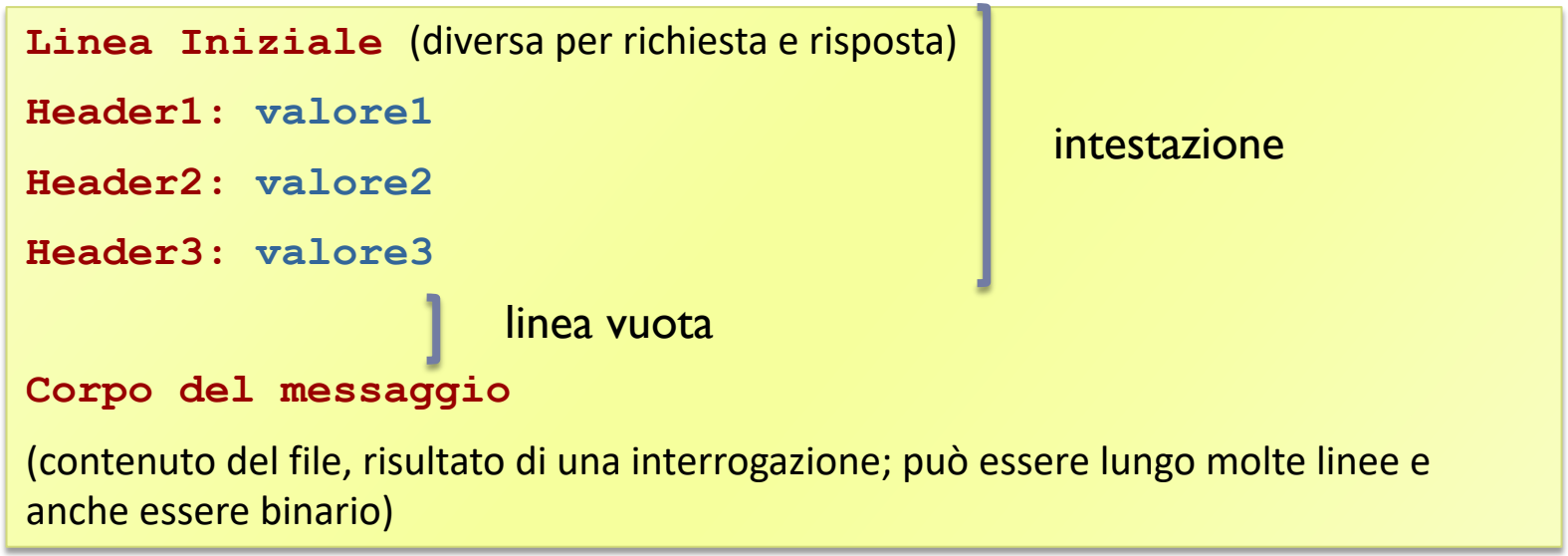
http https ftp mailto	nome DNS del server o indirizzo IP	percorso della risorsa & parametri (GET)

URI, URL & URN

- URI (Uniform Resource Identifier)
  - Necessità di identificare univocamente una risorsa sul Web
  - Ogni risorsa è identificata da una stringa di caratteri opportunamente formattata
  - URL (Uniform Resource Locator)
    - Rappresenta un indirizzo preciso che può essere utilizzato immediatamente per accedere ad una risorsa
    - Non è invariante rispetto alla rilocazione della risorsa
  - URN (Uniform Resource Names)
    - Nome assegnato stabilmente (permanentemente) ad una risorsa
    - E' indipendente dalla locazione fisica della risorsa
    - Deve essere associato ad un URL attraverso un servizio di naming
      - un esempio è il Digital Object Identifier (DOI) - *www.**doi**.org*
        - *http://dx.doi.org/10.1145/990301.990302*

Formato dei messaggi HTTP

- I messaggi HTTP di richiesta/risposta sono composti di due parti separate da una linea vuota
  - Intestazione (header): è un insieme di linee di testo con codifica ASCII e struttura standard
  - Corpo del messaggio (body): può contenere qualsiasi tipo di dato o essere anche vuoto



Richieste HTTP

- La linea iniziale di una richiesta è formata da 3 parti
  - Metodo (GET, POST, HEAD, PUT, DELETE, TRACE, OPTIONS, CONNECT, PATCH)
    - Indica la modalità della richiesta. Solo GET/POST sono obbligatori
  - Percorso locale della risorsa
    - E' la parte finale dell'URL dopo il nome dell'host
  - Versione di HTTP usata (HTTP/1.1 o HTTP/1.0)

```
GET /index.html HTTP/1.1
Host: univac.dii.unisi.it
User-Agent: Mozilla/5.0 (Windows; U; Win98; en-US; m18) Gecko/20010131
   Netscape6/6.01
Accept: */*
Accept-Language: en
Accept-Encoding: gzip,deflate,compress,identity
Keep-Alive: 300
Connection: keep-alive
```

Metodi HTTP

- I metodi definiscono l'operazione da effettuare sulla risorsa richiesta. I principali sono
  - GET
    Richiede il trasferimento di una risorsa. Se è seguita dall'intestazione **`If-Modified-Since`** il server invia i dati solo se sono stati modificati dopo la data specificata (gestione cache del browser)
  - HEAD
    Richiede solo le intestazioni relative alla risorsa. Serve per verificare le caratteristiche della risorsa senza trasferirla
  - POST
    Utilizzato per inviare dati da elaborare al server. L'intestazione è seguita da un corpo della richiesta che contiene i dati. Il tipo e la dimensione dei dati è indicata dagli header MIME **`Content-Type:`** e **`Content-Length:`**

Risposta HTTP

- La prima linea dell'intestazione riporta il codice dell'esito della richiesta (successo/errore)
  - Le intestazioni riportano informazioni sul server e sul contenuto del corpo della risposta (la risorsa)

header
informazioni sul server
&
sulla risorsa (formato)

risorsa
pagina base HTML

```
HTTP/1.1 200 OK
Date: Wed, 06 Jun 2001 22:44:40 GMT
Server: Apache/1.3.20 (Win32)
Last-Modified: Wed, 06 Jun 2001 22:32:26 GMT
ETag: "0-64-3b1eaf7a"
Accept-Ranges: bytes
Content-Length: 100
Connection: close
Content-Type: text/html

<HTML>
<HEAD>
<TITLE>Prova</TITLE>
</HEAD>
<BODY>
<H2>File HTML di Prova</H1>
</BODY>
```

Codici di stato 1xx e 2xx

- La linea di stato riporta un codice numerico e una stringa esplicativa
  - Il contenuto del corpo della risposta dipende dal codice
    - E' la risorsa richiesta se il codice indica successo
    - E' vuoto o contiene una pagina di errore se il codice indica insuccesso
  - I codici sono divisi in gruppi omogenei in base alla prima cifra
    - 1xx Informational
      - Indica che la risposta contiene solo l'intestazione
        - **`100 Continue`**
          Indica che il server ha ricevuto l'intestazione della richiesta e che il client può procedere ad inviare il corpo. E' usato per richieste di tipo POST che devono trasferire molti dati. Il client richiede questo tipo di conferma inserendo la linea di header **`Expect: 100-continue`** nella richiesta.
    - 2xx Success
      - Indica che la richiesta è stata ricevuta, interpretata ed eseguita correttamente
        - **`200 OK`**
          La risposta standard per richieste HTTP eseguite con successo

Codici di stato 3xx

- 3xx Redirection
  - Indica che il client deve eseguire un'ulteriore operazione per accedere alla risorsa
    - **`301 Moved Permanently`**
      - Questa e le richieste future devono essere redirette all'URI indicato nel campo **`Location:`** dell'intestazione
    - **`303 See Other`**
      - La risposta alla richiesta può essere trovata con una richiesta GET all'URI indicato nel campo **`Location:`** dell'intestazione. L'URI fornito non sostituisce quello richiesto

```
HTTP/1.1 303 See Other
Location: http://www.otherserver.net/otherresource
```

    - **`304 Not Modified`**
      - Indica che la risorsa non è stata modificata dall'ultima richiesta. E' utilizzata in risposta ad una richiesta che contiene il campo **`If-Modified-Since:`** (gestione cache/proxy)

Codici di stato 4xx & 5xx

- 4xx Client Error
  - Indicano casi in cui l'errore è attribuibile al client
    - **`403 Forbidden`**
      - La richiesta riguarda una richiesta per cui non si hanno permessi (es. Directory listing disabilitato)
    - **`404 Not Found`**
      - La richiesta fa riferimento ad una risorsa che al momento non esiste
- 5xx Server Error
  - Il server non riesce a rispondere ad una richiesta valida
    - **`500 Internal Server Error`**
      - Messaggio di errore generico (es. la risorsa richiede l'esecuzione di un programma che fallisce)
    - **`503 Service Unavailable`**
      - Il server non è al momento disponibile a rispondere alla richiesta per manutenzione o carico eccessivo

Campi dell'intestazione

- I campi dell'intestazione definiscono dei parametri operativi di una transazione HTTP
  - Sono linee di testo codificate come coppie **`Nome: valore`**
  - Le linee sono terminate dalla sequenza di caratteri CR LF
    - Campi lunghi possono essere spezzati su più linee; nelle linee dalla seconda in poi il primo carattere deve essere uno spazio o tab
  - Un insieme base di campi è stato standardizzato (RFC 2616) e devono essere implementati
    - Possono comunque essere utilizzati altri campi definiti (e usati) da applicazioni particolari
  - Lo standard non impone vincoli sulla dimensione dei singoli campi e sul loro numero
    - In pratica invece i server web/user agent possono avere limiti sulla dimensione massima (in byte) dell'intestazione e/o sul numero di campi

Esempi di campi della richiesta 1

- **User-Agent:**
  - Una stringa che identifica l'user-agent (il client/browser)
  - Specifica il nome e la versione del prodotto e del motore di layout
  - Può essere usata dal server per ottimizzare il formato della risorsa (es. layout della pagina) per una serie di user-agents predefiniti

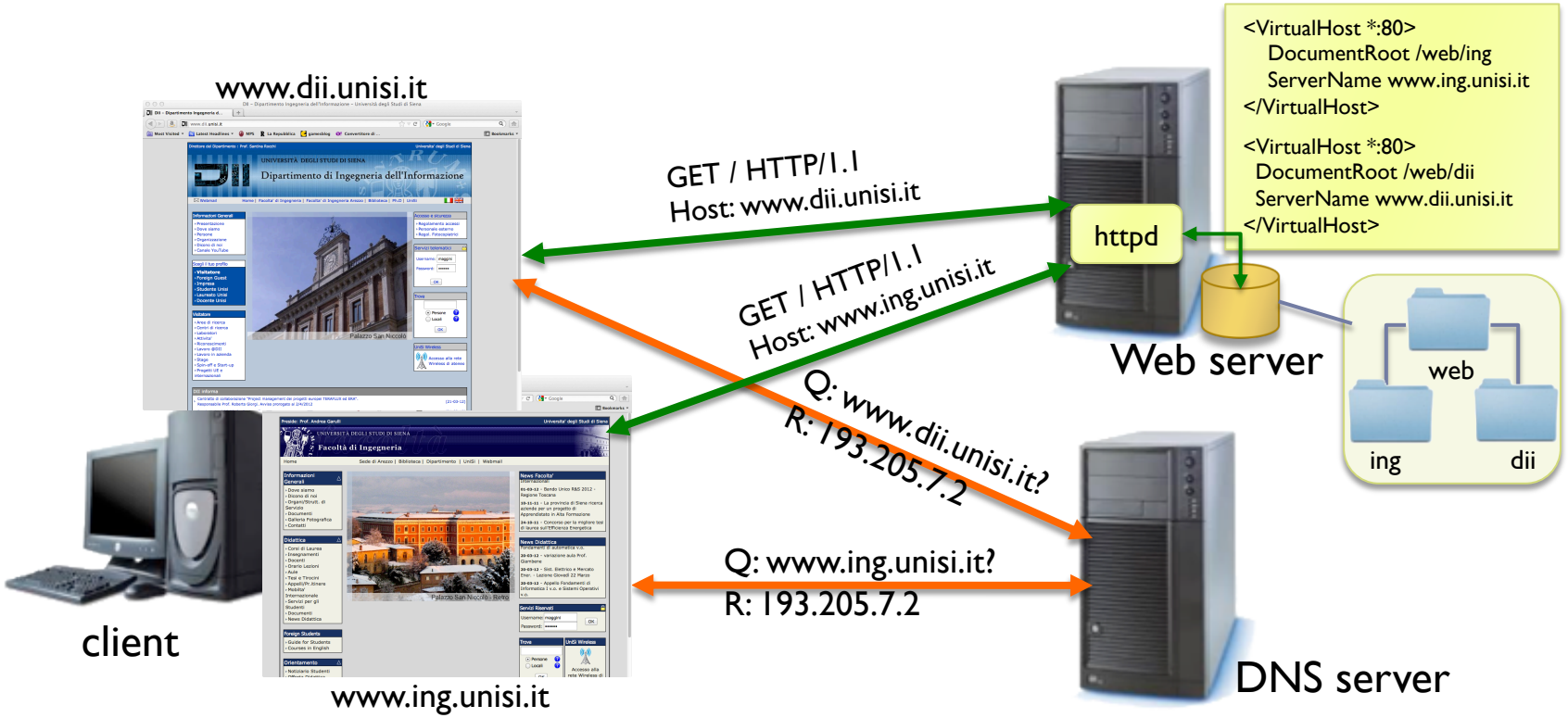
```
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_3)
  AppleWebKit/534.54.16 (KHTML, like Gecko) Version/5.1.4
  Safari/534.54.16
```

- **Host:**
  - E' un campo obbligatorio per HTTP/1.1 e riporta l'hostname a cui è diretta la richiesta (è l'hostname specificato nell'URL)
  - Permette la gestione del virtual hosting (più siti Web sullo stesso server con un unico IP)

```
Host: www.ing.unisi.it
```

(Name-based) Virtual hosting

- Il virtual hosting è una tecnica per ospitare più nomi di dominio su un singolo server
  - Permette di creare server Web condivisi (shared web hosting)
  - Il Web server utilizza il campo Host dell'intestazione per decidere quale sito web (virtual host) utilizzare



Esempi di campi della richiesta 2

- **`If-Modified-Since:`**
  - Serve per gestire il trasferimento condizionale alla ultima data di modifica della risorsa
  - Se la data di modifica sul server non è successiva a quella indicata, questo genera una risposta **`304 Not modified`** e non la trasferisce
  - E' stato introdotto per supportare la gestione della cache sia a livello di browser che di proxy

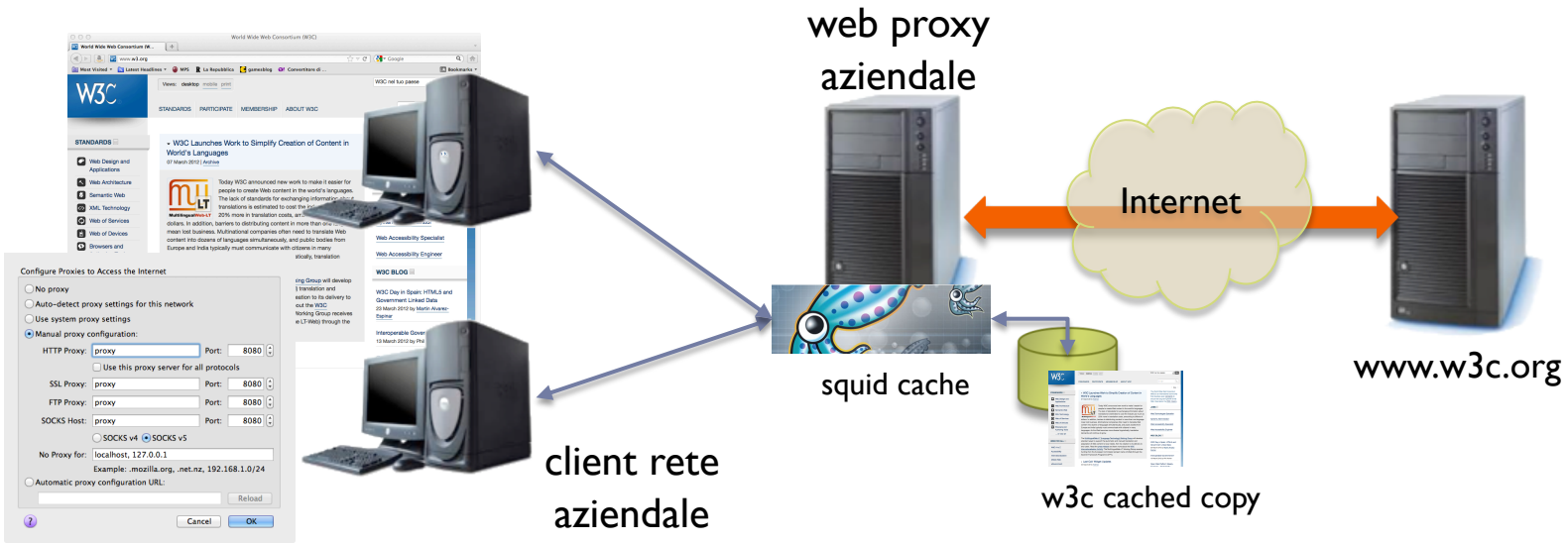
```
If-Modified-Since: Sat, 24 Mar 2012 20:38:22 GMT
```

- **`Cache-Control:`**
  - Permette di specificare delle direttive che devono essere applicate nella catena richiesta-risposta
  - E' possibile specificare la vita massima della risorsa (**`max-age=[seconds]`**), se la risorsa non è cachabile (**`no-cache`**), se la risorsa può essere cachata solo nel browser e non in proxy condivisi (**`private`**), ecc..

```
Cache-Control: max-age=3600, private
```

Web Proxy

- Un Web proxy è un intermediario per le richieste HTTP da un client verso i server Web
  - Riceve le richieste dal client, le propaga al server destinazione, recupera la risposta,e infine inoltra la risposta al client
  - L'uso del proxy Web è configurabile a livello di SO e/o di browser
  - Permette la gestione di caching condiviso, controllo di accesso e può essere l'unico canale per l'accesso al Web da una rete privata aziendale
  - Riduce l'uso della banda per l'accesso a risorse di interesse comune



Esempi di campi della richiesta 3

- **Accept:**
  - Indica una lista di tipi MIME che sono compatibili con l'user-agent
  - Si possono indicare tutti i tipi con la wildcard */*
  - In genere i browser possono gestire comunque i vari formati con plug-in, applicativi esterni o permettendo di salvare il file
    - Il comportamento dei browser è configurabile con l'associazione dei tipi
  - Il server può decidere di scegliere il formato della risposta in base ai tipi indicati dal browser

```
Accept: text/html
```

- **Accept-Language:**
  - Permette all'user-agent di specificare una lista di lingue per cui ha preferenza
  - Il server può inviare una versione localizzata della risorsa in base alla lingua indicata
  - Dipende dalla lingua in cui è installato il browser

```
Accept-Language: it, en
```

Esempi di campi della risposta

- **`Age:`**
  - Il tempo in secondi trascorso dalla risorsa in una cache

    `Age: 12`

- **`Content-Language:`**
  - La lingua in cui è la risorsa (se il server la specifca)

    `Content-Language: it`

- **`Content-Type: - Content-Length:`**
  - Il tipo MIME e la lunghezza in byte della risorsa
  - Il tipo MIME permette al brower di interpretare in modo corretto il formato della risorsa (per default in genere assume che sia **`text/plain`**)
  - Se la lunghezza della risorsa non è nota (contenuto generato dinamicamente) il server Web non specifica il campo **`Content-Length`** e utilizza una codifica "a pezzi" specificando **`Content-Encoding: chunked`**

    `Content-Type: image/gif`

    `Content-Length: 4578`

Caratteristiche HTTP

- Il protocollo HTTP è senza stato (stateless protocol)
  - Il server risponde alle richieste del client senza memorizzare nessuna informazione di stato sul client
  - Se il client invia più richieste queste sono di fatto indipendenti rispetto al meccanismo di HTTP
    - Se il client invia più volte la stessa richiesta il server risponde inviando lo stesso oggetto anche se non è stato modificato

- Più richieste possono essere gestite sulla stessa connessione persistente (HTTP/1.1 keep-alive)
  - Permette di gestire una sequenza di richieste verso lo stesso server utilizzando la stessa connessione TCP
  - Può essere disabilitata per mezzo del campo di intestazione `Connection: close` e viene comunque chiusa dopo un timeout dall'ultima richiesta (15s per Apache 2.0)
  - Riduce l'uso delle risorse

HTML (HyperText Markup Language)

- HTML è un linguaggio per descrivere la struttura delle pagine Web
  - Utilizza Markup Tags (annotazioni) per inserire comandi di formattazione di un testo
    - I Tag HTML sono keyword delimitate dai caratteri < >
    - In generale è previsto un tag di apertura e un tag di chiusura

```
<B>Testo in grassetto</B>
```

Testo in grassetto

    - Una coppia di tag individua un elemento della pagina
    - Il browser usa i tag per effettuare il rendering grafico della pagina
    - Per ogni tag possono essere specificati degli attributi che permettono di definire o modificare le proprietà del tag stesso
      - Per ogni tag sono predefiniti una serie di attributi con la relativa semantica e valori ammessi

```
<B style="color:red">Testo in grassetto rosso</B>
```

Testo in grassetto rosso

Struttura di un documento HTML

- HTML prevede la definizione di due parti del documento racchiuse fra i tag **<html>...</html>**
  - Intestazione **<head>...</head>**
    - Permette di specificare informazioni sul documento
      - Il titolo **<title>Titolo</title>** mostrato sulla barra del browser
      - Specifiche di stile con il tag **<style type="text/css">...</style>**
      - Codice di script usati lato client nella pagina **<script>..</script>**
      - Il riferimento a risorse esterne come fogli di stile (l'attributo **rel** specifica il tipo di relazione, **type** il tipo MIME, **href** l'URL della risorsa)

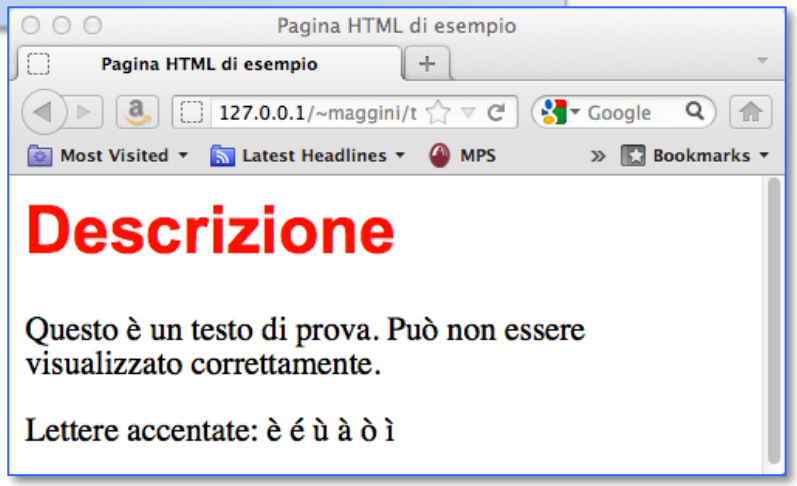
```
<link rel="stylesheet" type="text/css" href="site.css" />
```

      - Metadati associati al documento (keywords, author, intestazioni HTTP, charset) `<meta charset="UTF-8"/>`
  - Corpo **<body>...</body>**
    - Racchiude il contenuto testuale del documento insieme agli elementi (tag) necessari per la formattazione e l'identificazione delle parti

Esempio di pagina Web 1

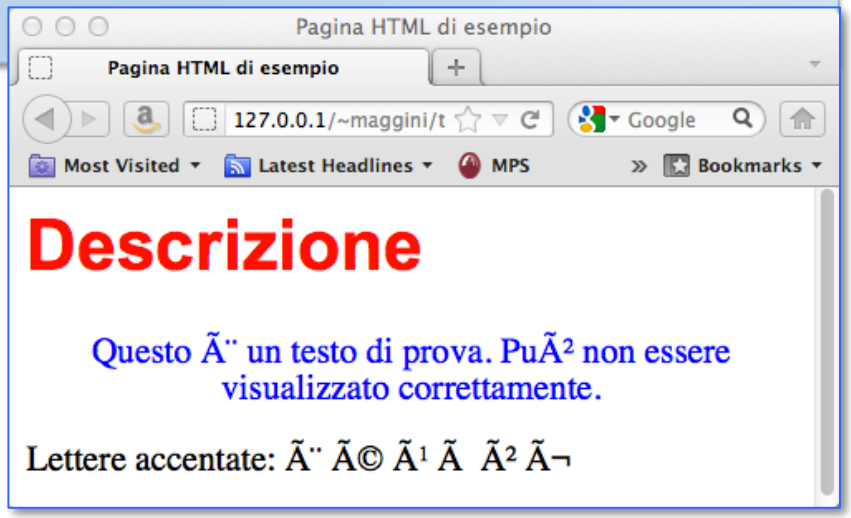
```
<html>
 <head>
  <title>Pagina HTML di esempio</title>
  <style type="text/css">
   h1 {color:red;font-family:arial}
  <meta charset= "UTF-8" />
 </head>
 <body>
 <h1>Descrizione</h1>
 <P>Questo è un testo di prova. Può non essere visualizzato
 correttamente.</P>
 <P>Lettere accentate: è é ù à ò ì</P>
 </body>
</html>
```

- l'intestazione definisce
  - Il titolo
  - lo stile di default (colore, font) per il tag **h1** (titolo di livello 1)
  - Il tipo di contenuto compreso la codifica dei caratteri usata (UTF-8)



Esempio di pagina Web 2

```
<html>
 <head>
  <title>Pagina HTML di esempio</title>
  <style type="text/css">
   h1 {color:red;font-family:arial}
   p.nice {color: blue; text-align:center}
  <meta charset="iso-8859-1" />
 </head>
 <body>
 <h1>Descrizione</h1>
 <P class="nice">Questo è un testo di prova. Può non essere visualizzato
 correttamente.</P>
 <P>Lettere accentate: è é ù à ò ì</P>
 </body>
</html>
```

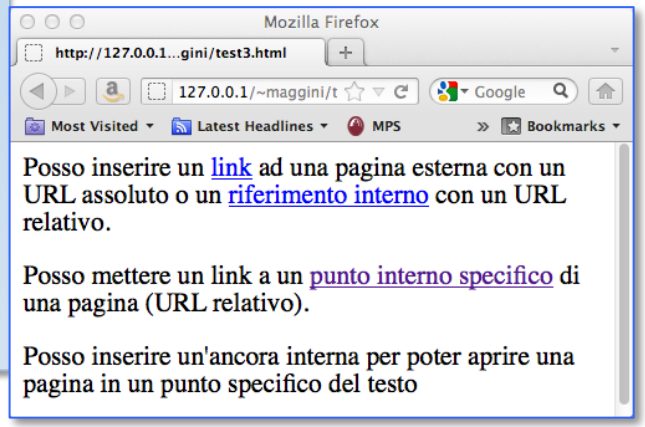


- E' stato inserito un charset errato (il testo è stato inserito con UTF-8)
  - indicando iso-8859-1 che è una codifica a lunghezza fissa su 8 bit si vede che i caratteri accentati hanno una codifica su 2 byte
- E' stato aggiunto uno stile per la classe di paragrafi "**`nice`**"

Ancore

- Ancora `<a>..</a>`
  - Permette di inserire un link nella pagina o un riferimento ad una posizione della pagina
    - Se specifica un riferimento, l'URL riferito è indicato come valore dell'attributo **`href`** (l'URL può essere assoluto - un URL completo o che inizia per / - o relativo alla pagina corrente)
      - Il testo fra i tag è il link evidenziato sulla pagina Web
    - Se marca una posizione interna nella pagina, l'attributo **`name`** definisce l'etichetta da usare nell'URL per puntare a questa posizione

```
<body>
 <p>Posso inserire un
 <a href="http://www.myserver.it/mydir/mypage.html">link</a>
 ad una pagina esterna con un URL assoluto o un
 <a href="../localpage.html">riferimento interno</a>
  con un URL relativo. </p>
 <p>Posso mettere un link a un
 <a href="test3.html#sec1">punto interno specifico</a>
 di una pagina (URL relativo). </p>
 <p> <a name="sec1" />Posso inserire un'ancora
  interna per poter aprire
 una pagina in un punto specifico del testo </p>
 </body>
```



Tag principali: immagini

- Inserimento di immagini <img />
  - Specifica il riferimento URL con il campo src
  - Con gli attributi si possono specificare le dimensioni (`width`,`height`) e una descrizione testuale (`alt`)
  - Con l'attributo `style` si possono specificare opzioni di visualizzazione (bordo, allineamento)

```
<body>
<h3 style="text-align:center">Un coniglio</h3>
<img src="images/rabbid.jpg" alt="rabbid" width=100
  style="border:5px ridge gray;padding:10px;margin:10px"/>
<br />
</body>
```

Tag principali: formattazione del testo

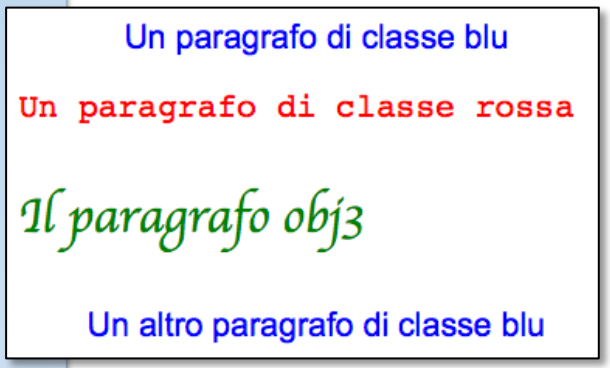
- HTML prevede tag per definire l'aspetto con cui è visualizzato il testo interno all'elemento
  - Ad esempio `<b>` grassetto, `<em>` italico, `<hn>` intestazioni di livello 1-6 ....
  - Alcuni sono sconsigliati dalla specifica HTML 4.01 perché è preferibile utilizzare l'attributo `style` (`<font>`, `<center>`)
  - Alcuni permettono di definire l'organizzazione del testo in paragrafi `<P>`, in liste ordinate `<OL>` e non ordinate `<UL>` di elementi `<LI>`, o di inserire un'interruzione di linea `<br \>`

```
<h4>Stili del testo</h4>
 <ul>
  <li><b style="font-family:arial">Grassetto</b></li>
  <li><em style="font-family:courier">Italico</em></li>
  <li><strong style="font-family:times">Strong</strong></li>
 </ul>
```

Attributi standard

- HTML definisce alcuni attibuti validi per tutti gli elementi
  - **class**
    - permette di assegnare un elemento ad una classe. E' utile per definire uno stile comune ad un sottoinsieme di elementi di un certo tipo (riferito con *.nome*)
  - **id**
    - permette di assegnare un identificatore specifico ad un dato elemento (riferito con *#id*)
  - **style**
    - permette di definire le proprietà di stile per l'elemento usando il formato dei CSS (cascading style sheet). Le proprietà sono ereditate da tutti gli elementi annidati.

```
<head>
<style type="text/css">
 p.blu {font-family:arial;color:blue;text-align:center}
 p.red {font-family:courier;color:red}
 #obj3 {font: italic 18pt cursive;color:green}
</style>
</head>
<body>
<p class="blu" id="obj1">Un paragrafo di classe blu</p>
<p class="red" id="obj2">Un paragrafo di classe rossa</p>
<p id="obj3">Il paragrafo obj3</p>
<p class="blu" id="obj4">Un altro paragrafo di classe blu</p>
</body>
```



Impaginazione con tabelle

- L'HTML ignora i caratteri di formattazione come spazi, tab e return
  - Per disporre i contenuti con un layout preciso si possono usare tabelle **<table>...</table>**
    - Le righe della tabella sono definite col tag **<tr>..</tr>** e le celle all'interno della riga con l'elemento **<td>..<\td>**
    - Con l'attributo **colspan** è possibile creare una cella che occupa più colonne e con **rowspan** una cella che occupa più righe
    - Gli attributi **cellspacing** e **cellpadding** permettono di definire la spaziatura orizzontale e verticare delle celle

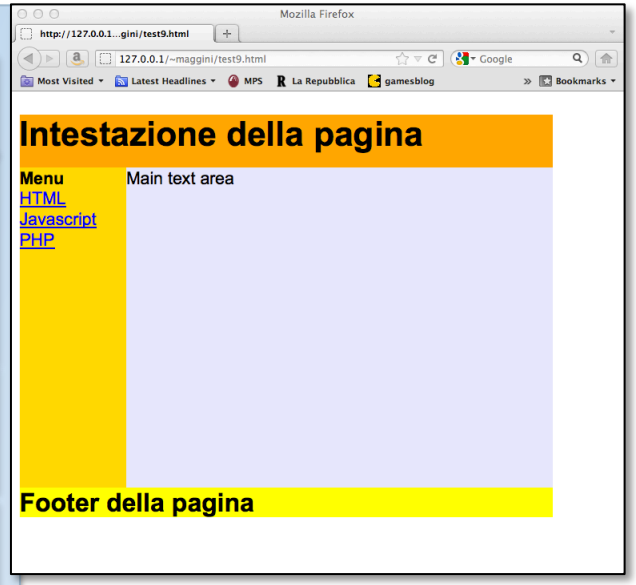
```
<head>
<style type="text/css">
 tr.head {color: #667788;background-color:#B0C4DE}
 tr.data {color: black;background-color:#FFE4E1}
</style>
</head>
<body>
<table style="text-align:center">
<tr class="head"><td>Nome e Cognome</td><td>Matricola</td><td>Voto</td></tr>
<tr class="data">
<td style="text-align:left">Mario Rossi</td><td>012345</td><td>18</td></tr>
<tr class="data">
<td style="text-align:left">Maria Bianchi</td><td>054321</td><td>21</td></tr>
<tr class="data">
<td style="text-align:left">Tim Barners-Lee</td><td>000001</td><td>30</td></tr>
<tr class="data">
<td colspan=2 style="text-align:right">Media</td><td>23</td></tr>
</table>
</body>
```

Nome e Cognome	Matricola	Voto
Mario Rossi	012345	18
Maria Bianchi	054321	21
Tim Barners-Lee	000001	30
	Media	23

div e layout

- L'elemento `<div>` rappresenta un blocco di altri elementi
  - Può essere usato per definire il layout
  - Permette di ottenere gli stessi risultati dell'uso di tabelle
  - Le regole di design delle pagine Web suggeriscono di usare questa soluzione invece di quella basata su `<table>`

```
<div style="width:500px;font-family:arial;">
  <div id="header" style="background-color:orange;height:50px">
    <h1>Intestazione della pagina</h1>
  </div>
  <div id="menu" style="background-color:gold;
     width:100px;height:300px;float:left">
    <b>Menu</b><br />
    <a href="html.html">HTML</a><br />
    <a href="javascript.html">Javascript</a><br />
    <a href="PHP.html">PHP</a>
  </div>
  <div id="content" style="background-color:lavender;
    width:400px;height:300px;float:left">
   Main text area
  </div>
  <div id="footer" style="background-color:yellow;clear:both">
    <h2>Footer della pagina</h2>
  </div>
</div>
```



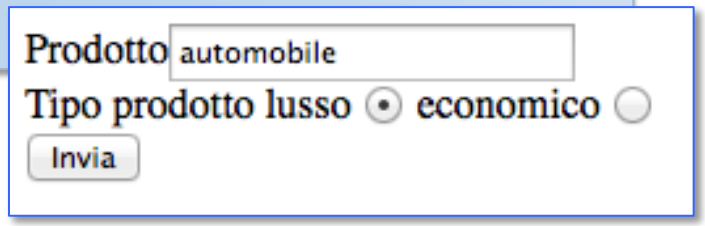
Passaggio di parametri in HTTP

- I servizi Web interattivi richiedono che il client possa inviare dati al server
  - In questo caso la risorsa richiesta al server Web è un'applicazione Web che viene eseguita per generare un contenuto Web dinamico
    - Il risultato dell'elaborazione è in genere una pagina Web che dipende dai parametri inviati ed eventulamente una modifica di stato sul server (es. memorizzazione di dati in un database)
  - I dati sono inviati nella richiesta HTTP e occorre definire:
    - Come sono codificati i dati in modo che il server possa interpretarli correttamente
    - Come sono rappresentati i dati nella richiesta HTTP
    - Come avviene la gestione a livello server dei dati (come vengono resi disponibili allo script o applicazione che deve fare l'elaborazione)

Form HTML

- Un modo per gestire l'inserimento in una pagina Web di dati da inviare al server è l'uso di Form HTML
  - Un form è un oggetto della pagina Web individuato dalla coppia di tag **<FORM>..</FORM>**
  - All'interno del form sono specificati gli elementi **<INPUT>** che permettono l'inserimento di dati da parte dell'utente

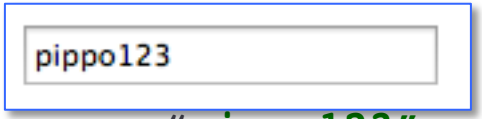
```
<FORM action="http://www.mmnect.com/cgi-bin/query.cgi" method=GET>
   Prodotto<INPUT name="prodotto" size=20 /><BR />
   Tipo prodotto lusso<INPUT name="tipo" type=radio value="lusso" />
   economico<INPUT name="tipo" type=radio value="economico" /><BR />
   <INPUT type=submit value="Invia"/>
</FORM>
```



Elementi <INPUT> e codifica dei dati

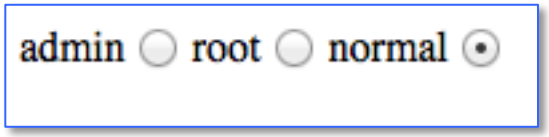
- Ad ogni elemento di INPUT è associato l'attributo **name** che definisce il nome di variabile a cui è associato il valore inserito per l'elemento
  - I parametri sono codificati come coppie nome="valore"
  - Il valore è una stringa di caratteri opportunamente codificata

```
<INPUT type="text" name="user">
```



user="pippo123"

```
admin<INPUT type="radio" name="role" value="admin">
root<INPUT type="radio" name="role" value="root">
normal<INPUT type="radio" name="role" value="normal" checked="true">
```



role="normal"

Tipi di Elementi INPUT

- Utilizzando l'attributo **type** è possibile variare il tipo di campo di input
  - **type=checkbox** definisce una scelta on/off
  - si può omettere l'attributo value che è la stringa "on" se il checkbox è selezionato

```
Interests<br />
<INPUT type="checkbox" name="movie">movies<br />
<INPUT type="checkbox" name="music">music<br />
<INPUT type="checkbox" name="arts">arts<br />
```

movie=on&arts=on

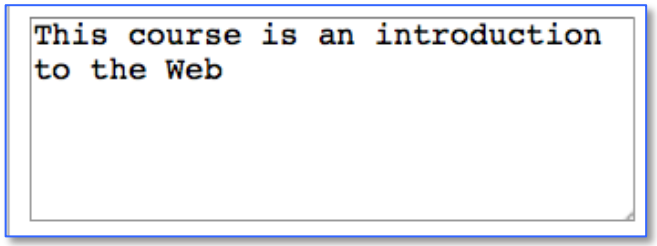
  - **type=submit** inserisce un bottone che attiva l'invio dei dati con una richiesta HTTP alla risorsa specificata nell'attributo action del tag <FORM>
  - L'attributo value può essere utilizzato per specificare l'etichetta sul bottone
  - Se si inserisce l'atttributo name viene creata la variabile corrispondente

```
<INPUT type="submit" value="Invia" >
```

Altri oggetti interattivi 1

- I form HTML possono contenere altri oggetti che permettono all'utente di valorizzare variabili da inviare al server
  - Il tag **<textarea>** definisce un campo di testo multiriga di lunghezza non limitata

```
<textarea cols=30 rows=5 name="abstract"></textarea>
```




```
abstract=This+course+is+an+introduction+to+the+Web
```

  - Può essere inserito un testo più lungo dello spazio visualizzato. Nel caso compare una scrollbar
  - Il valore assegnato alla variabile usa una codifica di tipo URL (url-encoded)

Altri oggetti interattivi 2

- Il tag `<select>` è utilizzato per creare menù a tendina
  - Le opzioni sono inserite col tag `<option>` che specifica la stringa da visualizzare
  - Se non è specificato esplicitamente con l'attributo `value`, il valore è la stringa contenuta nel tag `<option>` selezionato

```
<select name="genere">
  <option>Heavy Metal</option>
  <option>Death Metal</option>
  <option>Gothic Metal</option>
  <option>Hard Rock</option>
  <option>Progressive Rock</option>
</select>
```

`genere=Gothic+Metal`

  - Con l'attributo `multiple` si può definire una lista con scelte multiple

```
<select name="genere" multiple=true size=5>
```

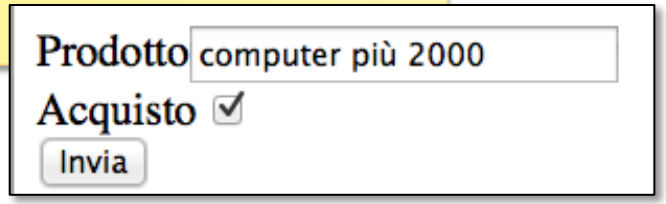
`genere=Heavy+Metal&genere=Gothic+Metal`

Invio dei dati con HTTP - GET

- I dati del form possono essere inviati con una richiesta HTTP di tipo GET
  - I parametri sono accodati all'URL della risorsa che li riceve usando in genere il carattere ? come separatore
  - Questo metodo permette di associare la richiesta con i parametri ad un URL (può essere usato come hyperlink, generato in automatico,..)
  - Esiste un limite al numero dei caratteri e inoltre l'URL deve essere composto da caratteri ASCII
    - I caratteri non ASCII e i caratteri speciali (?,=,&,..) sono codificati con la sequenza % seguito dalle due cifre esadecimali (es %E8 per è)
    - Lo spazio è in genere codificato con il carattere + invece che con %20
  - Il metodo GET non dovrebbe essere usato per passare informazione sensibile (es. password) perché è in chiaro nell'URL e può essere memorizzato nei proxy

Esempio di form e richiesta GET

```
<form action="/ecom/buy.php" method="get">
    Prodotto <input type="text" name="prodotto" size="20"><br/>
    Acquisto <input type="checkbox" name="compra"><br />
   <input type="submit" value="Invia">
</form>
```



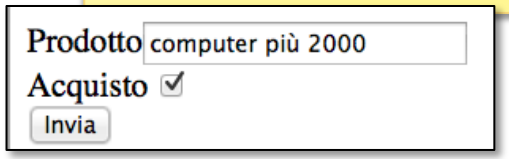

```
GET /ecom/buy.php?prodotto=computer+pi%F9+2000&compra=on HTTP/1.1
.....
```

- L'attivazione del form con il bottone **`submit`** genera una richiesta HTTP
  - L'attributo **`action`** specifica l'URL della risorsa a cui inviare i dati del form (nell'esempio un URL relativo)
  - L'attributo **`method`** indica che la richiesta è di tipo GET e che di conseguenza il client deve accodare i parametri all'URL nella forma URL encoded

Invio dei dati con HTTP - POST

- I dati del form possono essere inviati con una richiesta HTTP di tipo POST
  - I dati sono inseriti nel corpo della richiesta HTTP
  - Non si ha un riferimento URL alla richiesta con i parametri valorizzati
  - E' più robusto e sicuro del metodo GET e non ha limitazione sulla dimensione dei dati inviati

```
<form action="/ecom/buy.php" method="post">
   Prodotto <input type="text" name="prodotto" size="20"><br/>
   Acquisto <input type="checkbox" name="compra"><br />
  <input type="submit" value="Invia">
</form>
```



submit action

```
POST /ecom/buy.php HTTP/1.1
host: www.mycommerce.it
Content-type: application/x-www-form-urlencoded
Content-Length: 38

prodotto=computer+pi%F9+2000&compra=on
```

Encoding multipart/form-data

- L'attributo enctype permette di specificare il tipo di codifica con cui sono inviati i parametri di un form con metodo POST
  - Se non specificato la codifica è **`application/x-www-form-urlencoded`**
  - Se il form contiene un campo di input con **`type="file"`**, che permette di selezionare un file e inviarne il contenuto, occorre usare la codifica **`multipart/form-data`**
    - Utilizza il formato MIME multipart
    - Il messaggio contiene una serie di parti corrispondenti a ciascun elemento di input
    - Ciascuna parte ha un campo Content-Type: opzionale che per default è `text/plain`
    - Per ciascuna parte l'header Content-Disposition: specifica `form-data` e il nome del campo

Esempio multipart/form-data

```
<form action="http://myserver.it/archivia.php"
          enctype="multipart/form-data" method="post">

  Nome<input type="text" name="nome"><br />
  File<input type="file" name="file"><br />
  <input type="submit" value="Invia">
</form>
```


```
Content-type: multipart/form-data; boundary=--------------63341846741
Content-Length: 339

--------------63341846741
Content-Disposition: form-data; name="nome"

marco
--------------63341846741
Content-Disposition: form-data; name="file"; filename="pixel.gif"
Content-Type: image/gif

GIF89a
--------------63341846741
```

intestazione HTTP

corpo della richiesta HTTP

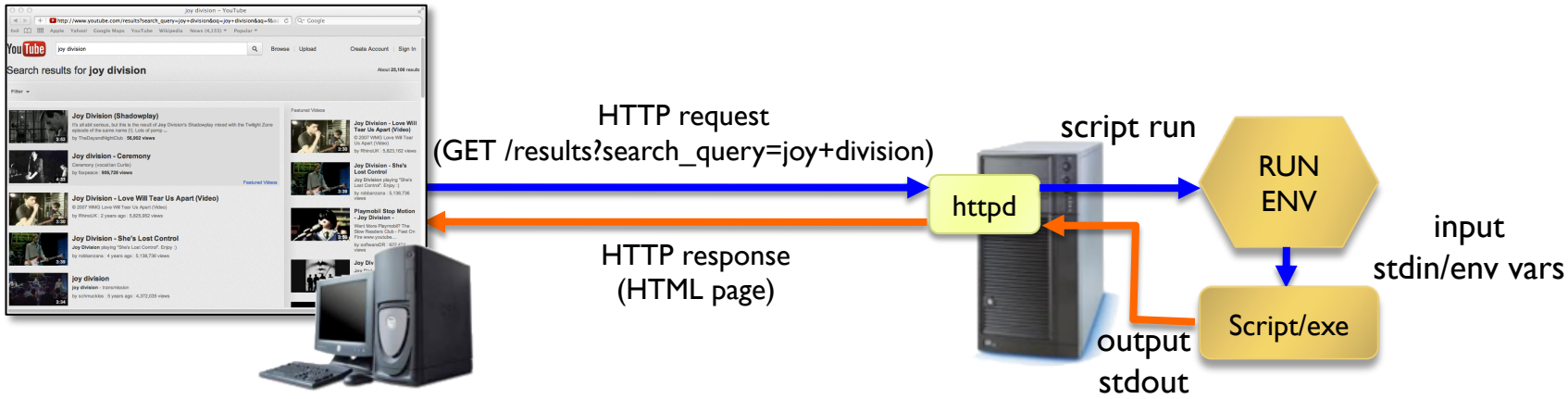
dati binari

Gestione di richieste “dinamiche”

- La risorsa che esegue l’azione individuata dall’URL specificato in un form deve eseguire un’elaborazione lato server
  - La risorsa è un programma (eseguibile o script interpretato) che deve essere eseguito dal server
  - I servizi sono riferiti e richiesti con la stessa modalità delle altre risorse statiche (URL e richiesta HTTP)
  - La richiesta HTTP a un servizio contiene (di norma) parametri, che servono all’elaborazione, codificati col metodo GET o POST
  - L’esecuzione del servizio produce dati che sono inviati dal server nella risposta HTTP
    - In genere è una pagina HTML, ma si possono generare altri tipi di dato come file pdf, immagini, ecc.. indicando il Content-Type opportuno
  - Il client non “vede” cosa è eseguito ma solo il risultato dell’elaborazione
  - Dal lato server è importante definire una modalità “portabile” con cui il programmatore può scrivere un servizio

Supporto del server Web per i servizi

- Un server Web generico deve poter supportare l'esecuzione di applicazioni collegate ad un URL
  - Deve capire in base alla risorsa richiesta se questa è un file già disponibile nel filesystem locale (es. pagina HTML, immagine) o se è un programma che deve essere eseguito
  - Deve fornire un supporto standard con cui interfacciarsi con l'esecuzione dell'applicazione "esterna" (lanciare l'applicazione, rendere disponibili i parametri, prelevare l'output per trasmetterlo al client)
  - L'insieme di informazioni che il server Web rende disponibili all'applicazione (es. parametri della richiesta HTTP) deve essere standard

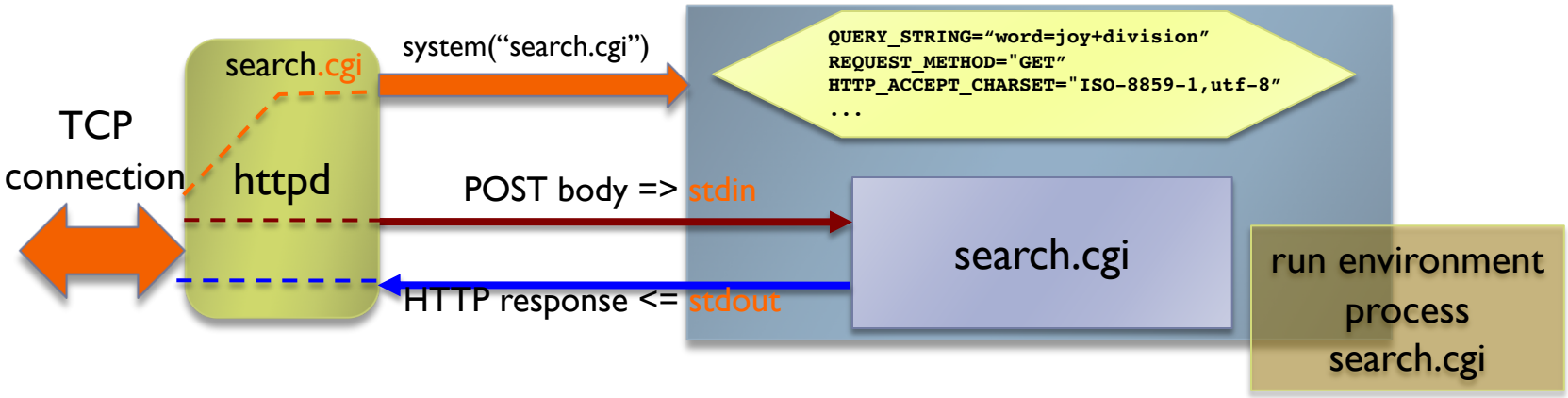


Gestione di eseguibili in un Web server

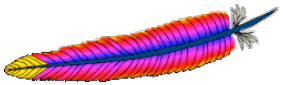
- Sono disponibili varie soluzioni per gestire l'esecuzione di eseguibili "esterni" ad un server Web
  - La possibilità di eseguire programmi di qualsiasi tipo rende il Web server flessibile
    - gli sviluppatori possono scrivere le applicazioni Web con il linguaggio e gli strumenti preferiti (C, C++, Java, perl, phyton,...)
  - Esiste un problema di efficienza per gestire un alto carico di richieste
    - Diventa critica la modalità con cui è realizzata l'esecuzione del programma esterno e come sono passati i dati della richiesta HTTP
  - Ad esempio lo standard CGI (Common Gateway Interface) indica una modalità di interazione fra un server Web e script eseguibili
    - Non è una soluzione particolarmente efficiente perché prevede l'esecuzione di un processo esterno per ogni richiesta e la procedura per l'esecuzione può essere più costosa di quella dell'effettiva generazione della risposta

CGI in un Web server

- La risorsa richiesta è identificata dal server web come un programma in base all'estensione del nome del file o alla directory
- Il server HTTP predispone l'esecuzione del programma CGI definendo l'ambiente di esecuzione assegnando i valori dei parametri che caratterizzano la connessione col client a variabili di ambiente
- Il server HTTP reindirizza l'input e l'ouput della connessione TCP del client sullo stdin e stdout del programma CGI
- Il server esegue l'applicazione che invia sul suo stdout la risposta HTTP (intestazione e corpo del messaggio) e chiude la connessione quando il programma CGI termina



Apache mod_cgi



- Si può associare la gestione come CGI associando i file al gestore di cgi-script nel file di configurazione del server Web (`httpd.conf`)
  - in base all'estensione del nome del file (esempio .cgi .pl - perl)

```
AddHandler cgi-script cgi pl
```

  - in base alla directory in cui si trova il file

```
ScriptAlias /cgi-bin/ /web/cgi-bin/
```

- Per motivi di sicurezza è necessario abilitare l'esecuzione di script cgi per ogni directory (l'abilitazione è ereditata dalle sottodirectory)

```
<Directory /web/htdocs/specialdir>
  Options +ExecCGI
</Directory>
```

  - Il programma CGI deve essere eseguibile sul SO dall'utente con cui è eseguito httpd (permesso +x su UNIX)

Programmazione server-side

- Sono disponibili soluzioni più efficaci ed efficienti per la programmazione server-side
  - Supporto specifico per la programmazione Web server-side
    - accesso diretto alle informazioni della richiesta HTTP con variabili
    - semplificazione della generazione dell'output (HTML embedded, manipolazione delle intestazioni HTTP)
  - Migliore interazione con il server Web
    - Ottimizzazione dell'esecuzione e del passaggio dei parametri
    - Ad esempio moduli plug-in per l'esecuzione degli script in Apache
      - lo script è eseguito da un modulo del server Web senza generare un processo esterno e la comunicazione usa metodi più efficienti
      - Per il PHP in `httpd.conf`

        ```
        LoadModule php5_module libexec/apache2/libphp5.so
        AddType application/x-httpd-php .php
        ```
    - Implementazione di server Web orientati all'esecuzione di applicazioni Web (application server) in uno specifico linguaggio
      - Java: Tomcat, IBM Websphere, GlassFish,... PHP: Zend PHP server