MODULO DI GIS SU WEB

Master in Sistemi Informativi Territoriali , AA 2009/2010

Chiara Renso KDDLAB, ISTI- CNR, c.renso@isti.cnr.it

Contenuti del Corso

Il corso è suddiviso in tre parti:

1. Introduzione e concetti di tecnologia web
    a. Internet e Tecnologia Web
    b. Il server web: IIS, Apache, IIS
    c. Il Linguaggio HTML
    d. Tecnologie Web Server-Side e Client-Side
2. Applicazioni WebGIS
    a. I servizi WebGIS
    b. Strategie di implementazione delle applicazioni WebGIS
    c. Gli standard OpenGIS
3. I prodotti WebGIS
    a. Prodotti WebGIS commerciali e opensource
    b. Approfondimento ed esercitazione su MapServer by UMN

Introduzione

Questo corso si propone di fornire un'introduzione alle tecnologie di supporto alle applicazioni di sistemi GIS su web. Il corso si articola in parti. Una prima parte vedremo una introduzione su alcuni concetti di base di tecnologia web, concetti necessari alla comprensione delle applicazioni webgis. Poi vedremo nel particolare quali sono le problematiche di web mapping, infine una breve panoramica sui prodotti esistenti e un approfondimento su un prodotto OpenSource, MapServer, (http://mapserver.umn.edu).

Cosa significa WebGIS

Con il termine WebGIS si indica un insieme di tecnologie che permettono di sfruttare le funzionalita' GIS via Web (Internet/Intranet).

Altri termini analogalmente usati sono Web-based GIS, Online GIS, Distributed GIS e Internet Mapping.

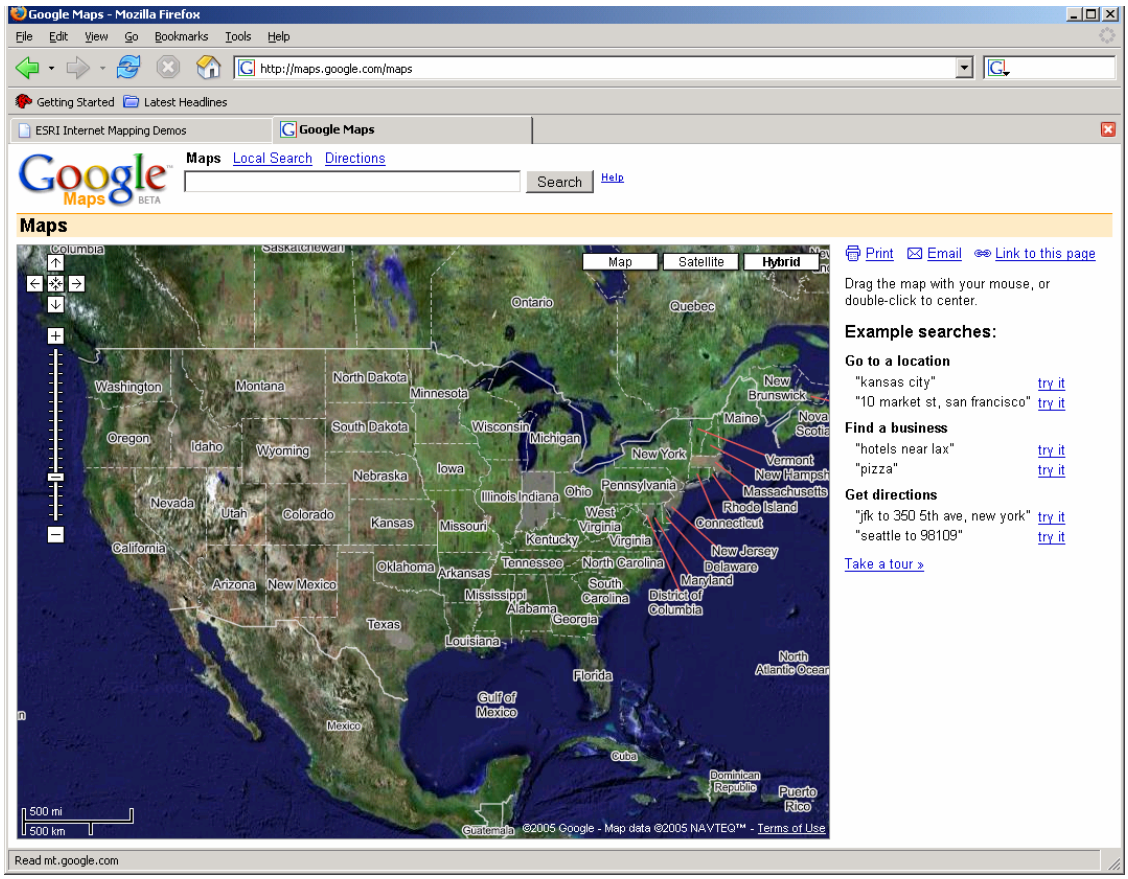
Con questo termine in senso ampio si intendono sia servizi Internet/Intranet, che specifici pacchetti software commerciali e non, usati per sviluppare questi servizi. In particolare, si intendono tutte le tecnologie di base e gli standard che sono stati sviluppati per rendere questi servizi possibili, istituzioni e iniziative che sono state create per incoraggiare lo sviluppo di queste tecnologie e questi standard e la ricerca scientifica sia nel campo sociale, cognitivo e tecnico che nasce dall'uso di queste tecnologie.

Vedremo durante il corso che le cosiddette *Applicazioni WebGIS* possono andare dalle semplici mappe statiche su una pagina web, a servizi più sofisticati di analisi fino ai modelli di GIS collaborativo network-based nel quale utenti in locazioni remote condividono dati comuni e comunicano gli uni con gli altri in tempo reale.
Le tecnologie che si stanno sviluppando per rendere le applicazioni WebGIS possibili includono software per i servers (per gestire dati e applicazioni), per i clients (che usato i dati e le applicazioni) e la comunicazione su rete (che controlla il flusso di informazione tra server e client).
I vantaggi di usare tecnologia WebGIS sono molteplici, tra cui il basso costo, la facilità d'uso, la grande diffusione di Internet, la vasta diffusione di dati geografici.
Le applicazioni di GIS su Internet tendono a realizzare su pagine web visualizzate dal browser le funzionalita' dei sistemi geografici, generalmente di *complessità molto meno elevata* di un pacchetto GIS.
Ogni applicazione si puo' presentare in modo completamente differente dalle altre, in base, non solo alla grafica realizzata, ma anche alle *strategie* adottate e ai *servizi* offerti.

Alcune applicazioni realizzano su web dei veri e propri GIS con funzionalita' di *zoom*, *pan*, ma anche visualizzazione di *layers*, visualizzazione dei attributi tematici, funzionalita' di analisi.

Le applicazioni webgis piu' semplici e comunemente piu' usate sono le visualizzazioni delle mappe interattive delle citta'. Qualche esempio più comunemente usato sono i siti di Viamichelin (http://www.viamichelin.it) e Google Maps (http://maps.google.it).

Recentemente si è riscontrato molto interessa per l'applicativo Google Earth, che rappresenta un "*mappamondo sul PC*". E' possibile posizionare il puntatore del mouse su qualunque luogo del pianeta che si desideri, ed ottenere informazioni su indirizzi, hotel, luoghi da visitare. Google Maps è la versione web based di google earth:

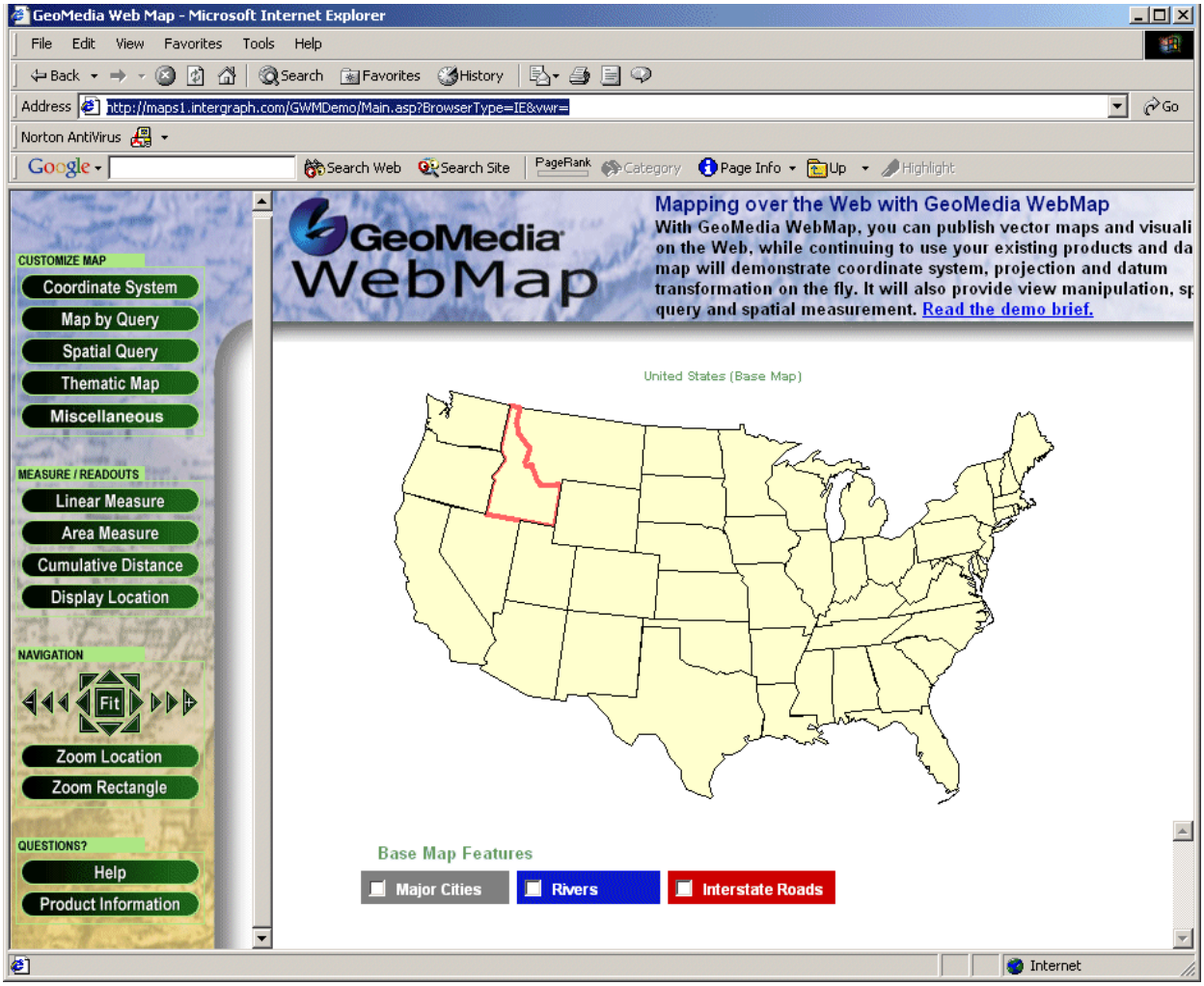


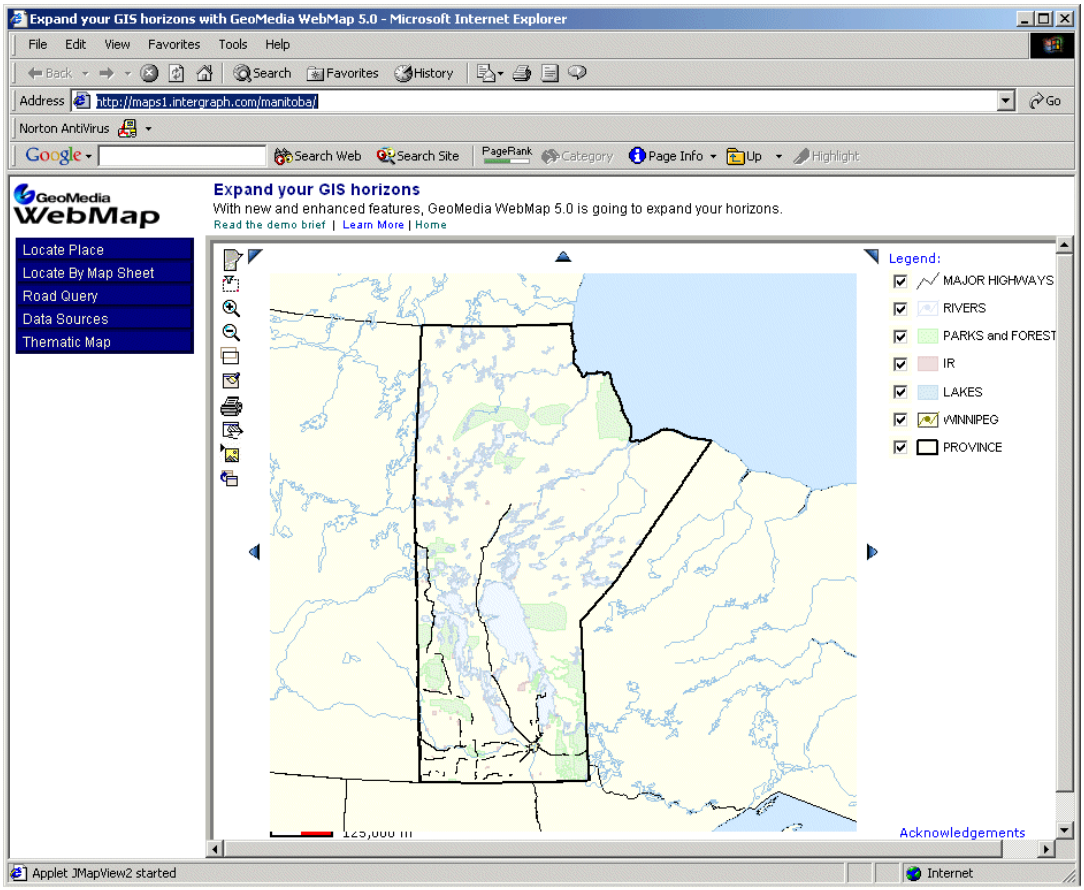
Applicazioni WebGis piu' complesse realizzano funzionalita' GIS quali visualizzazione di layers e funzionalita' di analisi. Alcuni esempi sono:

http://earth.jscc.ru/tenerif/?lang=en

http://resources.esri.com/showcase/

http://maps1.intergraph.com/manitoba/





Lo scopo di WebGIS è creare sistemi software che siano *platform-independent* e aperti ad ogni computer connesso ad Internet e su cui e' in esecuzione un server Web.
Le figure professionali coinvolte nello sviluppo/uso di siti WebGIS vanno da specialisti GIS a specialisti di sistemi informativi, webmaster, manager, marketing e addetti public relations.
Vedremo che progettare una applicazione WebGIS significa, non solo la realizzazione dal punto di vista tecnologico, ma anche porsi il problema di *quale e' l'audience* che ci aspettiamo, chiedendoci ad esempio se gli utenti saranno esperti o meno di GIS e/o di web, quali sistemi avranno (connessioni dirette veloci, computer potenti...), quali sistemi GIS possiedono, dove saranno collocati (interni all'organizzazione, a livello nazionale, internazionale.
WebGIS è un valore aggiunto per un sito web, in quanto permette l'uso di dati di tipo spaziale fornendo cosi' vari tipi di servizi quali l'accesso a informazione spaziale, service delivery, l'accesso ai dati e il data commerce. Inoltre, può essere considerato valore aggiunto anche per i dati stessi che una volta pubblicati su web possono essere facilmente acceduti dagli utenti senza la necessità di usare un software GIS.

Utilizzare applicazioni WeGIS significa cercare un compromesso tra i vantaggi ma anche gli eventuali svantaggi che tale applicazione può avere al fine di trovare un compromesso ottimo tra i requisiti dell'applicazione e le difficoltà della tecnologia utilizzata.

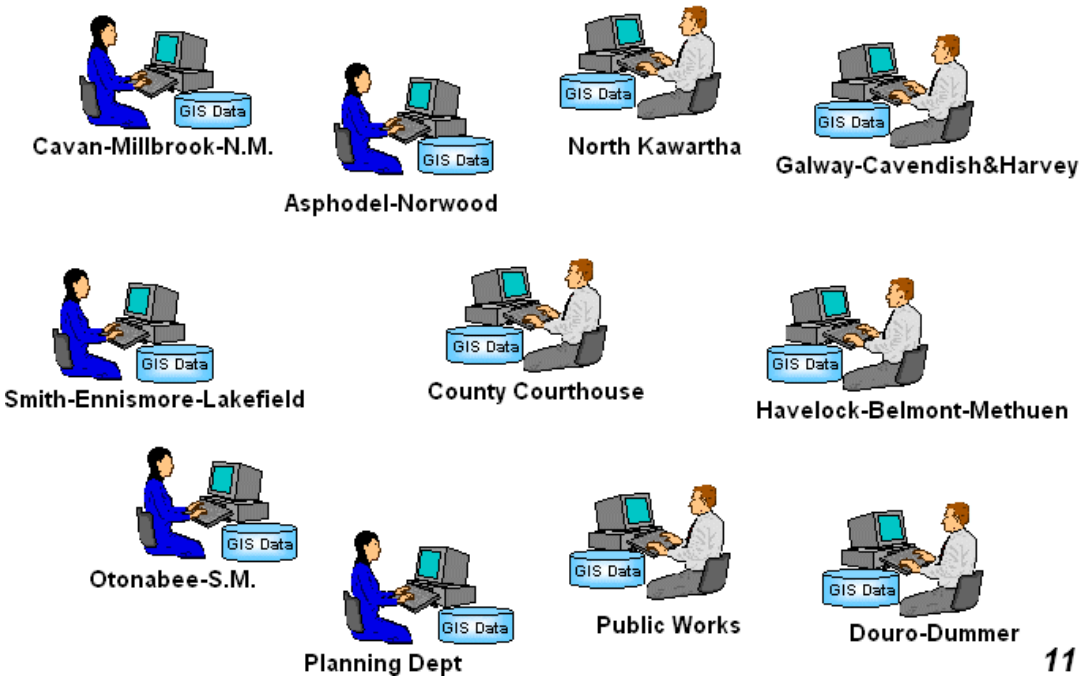
Principali Vantaggi

- Un singolo dataset centralizzato
- HW e SW a basso costo
- Facilità d'uso
- Basato su browser web, quindi necessita di minimo addestramento
- Ampio accesso a funzionalità GIS
- Integrazione di dati provenienti da fonti diverse
- Posso permettere la navigazione e visualizzazione dei dati senza dare i dati sorgenti

Svantaggi

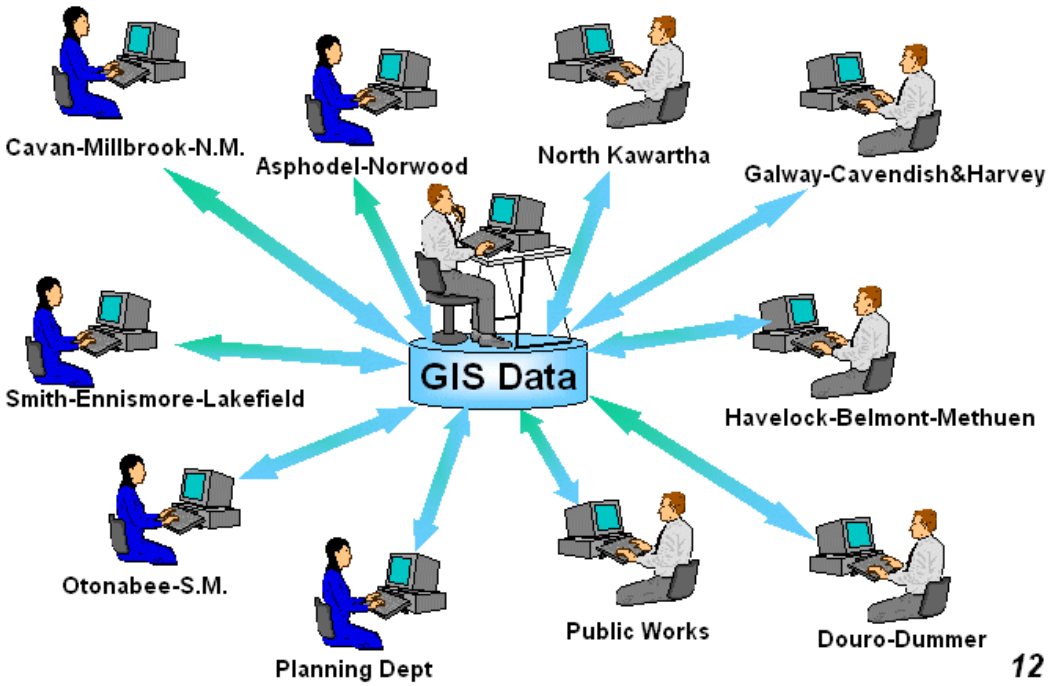
- Il tempo di risposta può essere relativamente lento e dipende:
- dal server
- dal client
- dalla banda di connessione Internet
- dal traffico del sito e della rete
- efficienza nei dati
- la applicazioni web generalmente non offrono tutte le funzionalità di un GIS desktop

Conventional Distributed GIS



11

Web-based GIS



12

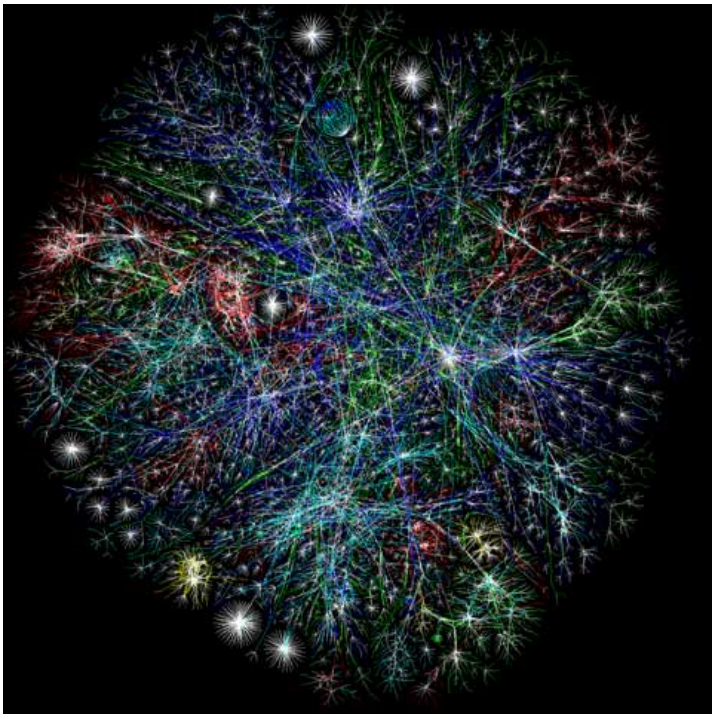
Vedremo nel corso come la tecnologia web può essere sfruttata per applicazioni di tipo geografico. Lo scopo e' quello di sfruttare il browser web, che ha una larga diffusione tra gli utenti a costo praticamente zero e la capillarita' di Internet, per portare applicazioni geografiche ad un pubblico sempre piu' vasto.
La struttura che intende rendere pubblici i propri dati geografici deve decidere sia che tipo di *servizio* vuole offrire, sia che tipo di *strategia* adottare e quindi anche che *tecnologia* usare, eventualmente decidere la spesa e l'effort in termini di risorse umane che è può preventivare (commerciale/opensource) per costruire la propria applicazione.

INTRODUZIONE A INTERNET E TECNOLOGIA WEB

Internet e' una collezione globale di reti gestite congiuntamente da organizzazioni private, università e agenzie governative.

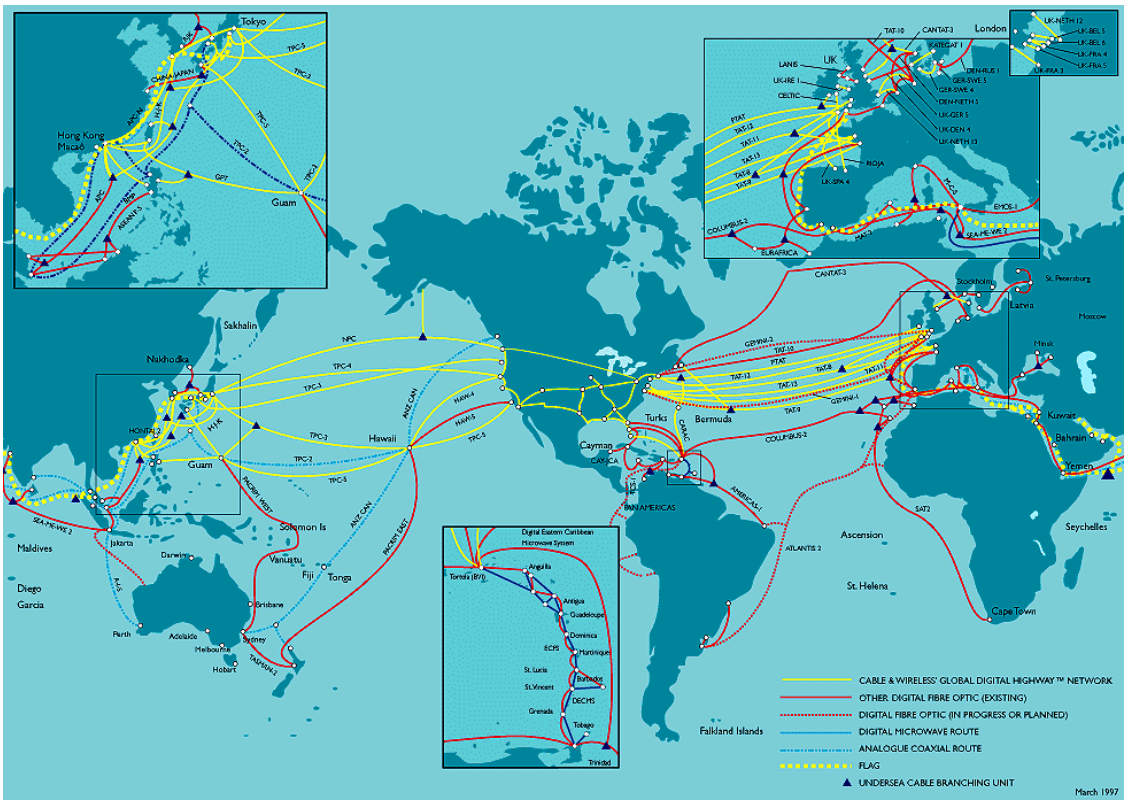
Internet può avere diverse definizioni. Può essere vista come una struttura globale di connessione di reti di computer, uno strumento di comunicazione, una rete di persone collegate attraverso computer, un villaggio globale

Non ha un *proprietario,* ma nasce e vive dalla collaborazione di tutti



Un'immagine della connettività di internet

Si chiamano reti *Intranet* reti locali che usano la tecnologia Internet. Una rete *Extranet* e' una rete privata (una organizzazione, un ente) su area geografica.
La connessione a livello mondiale si ottiene tramite delle linee a larga banda che collegano gli stati o i continenti tramite cavo sottomarini.



Mappa Mondiale Internet Backbone

Il numero di utenti internet cresce esponenzialmelmente nel tempo. Questa classifica è riportata nel CIA World Factbook[1] e vede la Cina come il paese in assoluto con più utenti. L'Italia risulta al dodicesimo posto.

Rank	Country	Internet users	Date of Information
1	**World**	1,018,057,389	2005
2	**China**	253,000,000	2008
3	**European Union**	247,000,000	2006
4	**United States**	223,000,000	2008
5	**Japan**	87,540,000	2006
6	**India**	60,000,000	2005

[1] https://www.cia.gov/library/publications/the-world-factbook/rankorder/2153rank.html

7	**Brazil**	42,600,000	2006
8	**Germany**	38,600,000	2006
9	**Korea, South**	34,120,000	2006
10	**United Kingdom**	33,534,000	2006
11	**France**	31,295,000	2007
12	**Italy**	28,855,000	2006
13	**Russia**	25,689,000	2006
14	**Canada**	22,000,000	2005
15	**Mexico**	22,000,000	2006

TCP/IP

Internet e' un mezzo di comunicazione tra computer. Affinché due entità comunichino occorre definire delle **regole di comunicazione dette** *protocolli***.** Tali regole specificano come codificare il segnale, in che modo far viaggiare i dati etc... L'insieme di protocolli che permette la comunicazione su Internet e' chiamato *TCP/IP*.

E' un protocollo *indipendente dalla rete fisica* (fibra ottica, cavo telefonico, rete locale..), si basa sul concetto di *pacchetto* ed e' un *open standard*.

Protocollo di comunicazione tra computer

Schema di funzionamento del protocollo TCP/IP:

Quando l'informazione (msg email, pagina web…) deve essere trasferita sulla rete:

1. Viene spezzettata in pacchetti contenenti l'indirizzo destinatario.
2. Ogni pacchetto separatamente viene spedito sulla rete e puo' prendere una strada diversa dagli altri (reti a commutazione di pacchetto).

3. Il destinatario riceve tutti i pacchetti, li riordina e li riassembla

Schema funzionamento TCP/IP

Ogni computer collegato ad Internet e' univocamente identificato dal suo *IP address*, una serie di 4 cifre da 1 a 255 separate dal carattere "punto" (.).

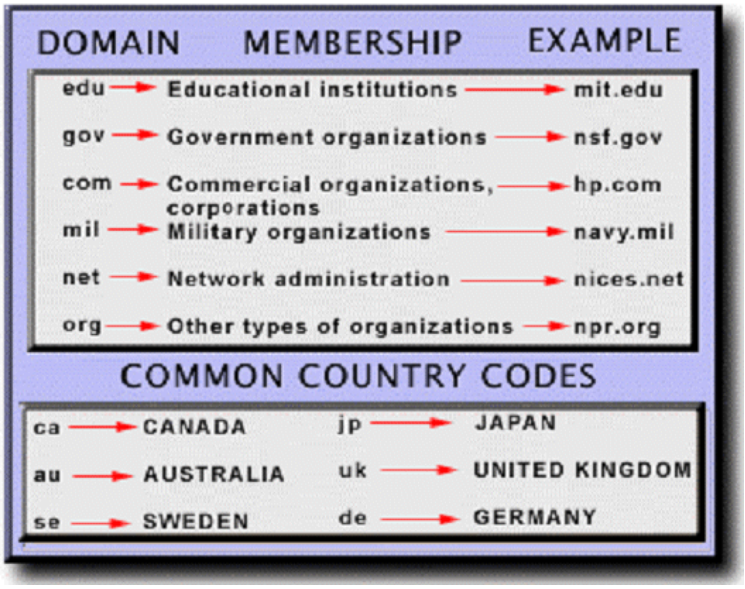
Un esempio di IP Address:

`146.48.82.93`

Ad un IP address puo' essere associato un *nome logico*, con il quale generalmente ci riferiamo navigando sul Web:

`www-kdd.isti.cnr.it`

Un nome logico e' formato da una parte che identifica **il dominio** (il nome della sottorete locale) e una parte che identifica il computer all'interno del dominio. Nell'esempio il dominio e' `isti.cnr.it`, il nome del computer e' `www-kdd`.
Il suffisso identifica il paese di appartenenza del dominio (`.it` per l'Italia) oppure i siti commerciali (`.com`), universitari (`.edu`), governativi (`.gov`) o militari (`.mil`). Numerosi altri suffissi di primo livello sono disponibili, come .biz, .name, .info etc



Il meccanismo che traduce da nome logico (piu' leggibile e mnemonico per l'utente) a IP address (riconosciuto dal protocollo IP) si chiama *Domain Name Server* (DNS).

Alcuni nodi della rete (computer generalmente dedicati a DNS) strutturati in modo gerarchico si occupano della traduzione degli indirizzi. In questo modo possiamo riferirci ad un computer nella rete sia tramite il suo indirizzo IP che tramite l'indirizzo logico.

Come Collegarsi a Internet

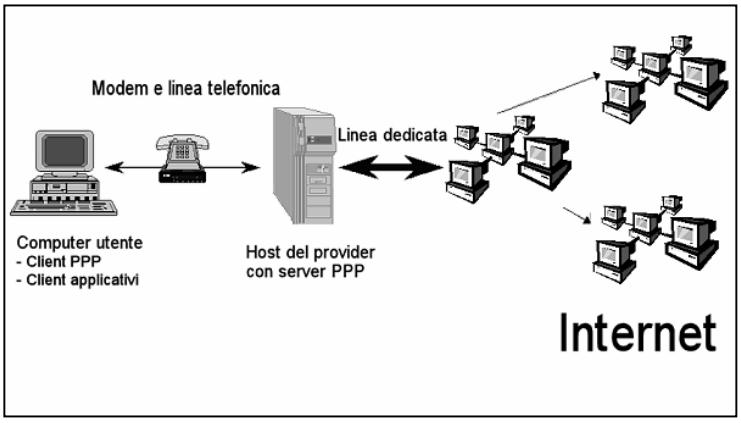
Ci sono due possibilità principali: una è tramite linea dedicate, l'altra è tramite un provider.

Linea Dedicata

- Grande aziende o enti
- Investimento iniziale consistente
- Non si paga il traffico ma solo l'accesso
- Si richiede l'accesso Internet ad un provider che ci assegna un insieme di indirizzi IP
- Due figure:
- Fornitore di accesso - IP - (Infostrada, Telecom, ….)
- Fornitore di linee - cavi - (Telecom, ….)

Collegamento temporaneo

- Collegamento tra due computer tramite linee telefoniche
- Meno costoso inizialmente, si paga il tempo di connessione (bolletta telefonica) o il canone ADSL
- Serve un modem collegato al PC che si vuole connettere e un provider che ci fornisca l'accesso (Wind, Tiscali, Infostrada, fastweb,…)
- L'accesso e' gratuito ma paghiamo la telefonata o il canone per l'ADSL
- Si usa il protocollo PPP (TCP/IP su linea commutata)



I Servizi di Internet
Internet e' una struttura di rete sulla quale vengono realizzati vari servizi di livello applicativo.
Esempi di servizi forniti da Internet sono*, ftp, email, WWW, chat, skype*. Ognuno di questi servizi comunica tramite protocolli basati su TCP/IP. Ad esempio la posta elettronica usa il protocollo SMTP, le news NNTP, il WWW il protocollo HTTP.

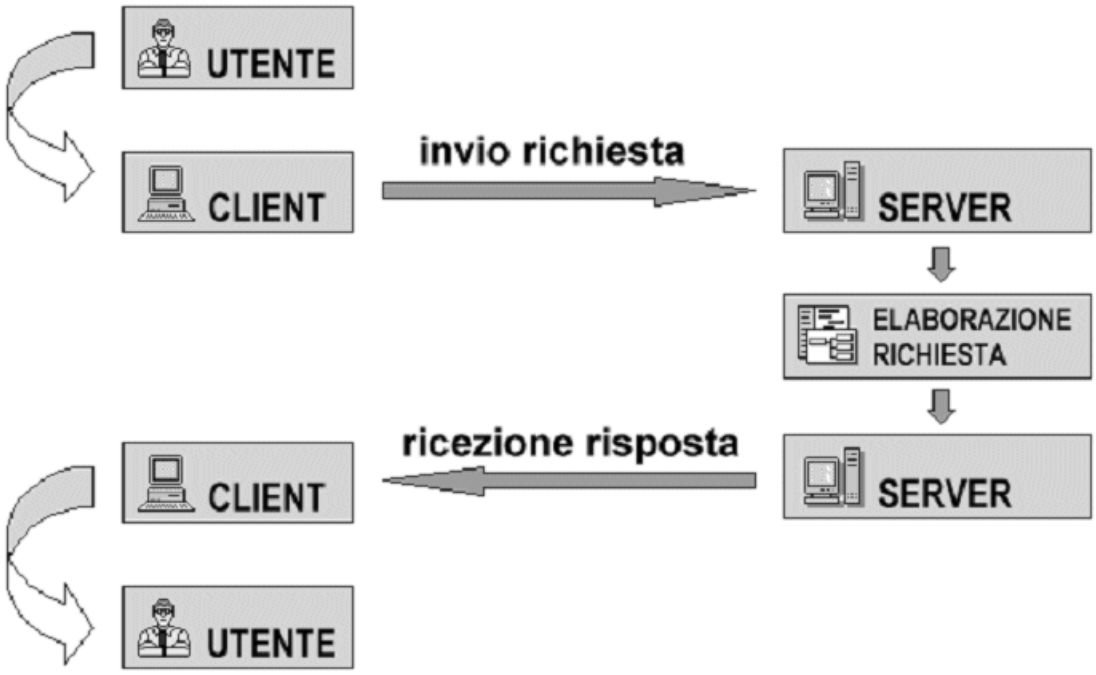
In questo corso ci concentriamo sul servizio WWW (web).

Architettura Client-Server

La comunicazione su Internet si basa su un'architettura *Client-Server*.
In un'architettura Client-Server due moduli indipendenti interagiscono per eseguire un compito.

L'idea è che il cliente richiede l'esecuzione di un servizio al servente il quale lo esegue e ritorna la risposta al cliente.

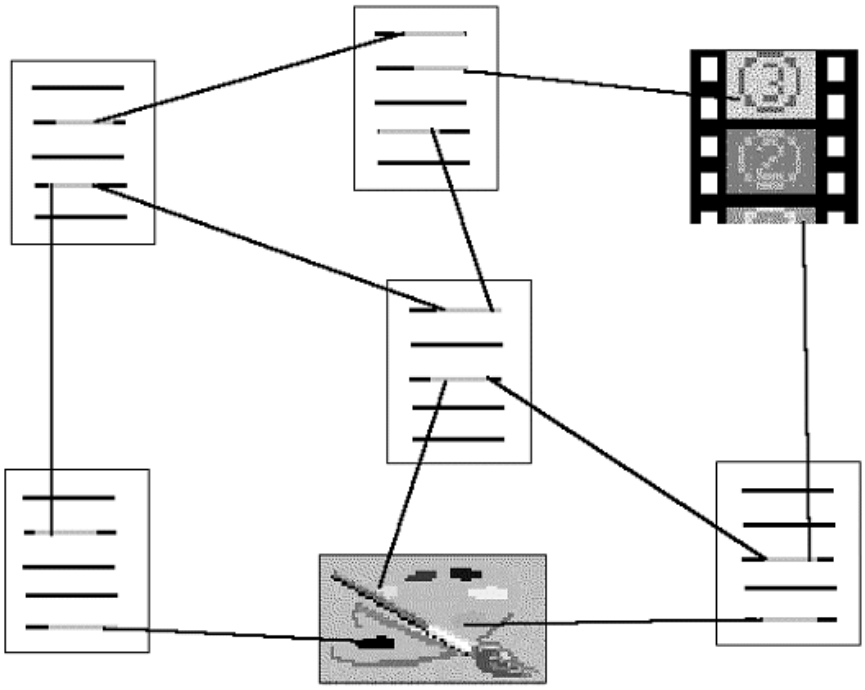
Un computer collegato alla rete è un **server** per un particolare servizio se gira il programma che fornisce quel servizio. Uno stesso computer puo' essere un servente per un servizio e un cliente per un altro .
Su Internet il cliente e il servente sono due computer collegati alla rete e il servizio richiesto dipende dal particolare servizio di Internet che stiamo richiedendo (ftp, www, msn...).
Il client è il generico utente che può accedere ad altri computer della rete attraverso il web browser (Firefox, Opera, Internet Explorer).
Il browser manda una richiesta al server web in esecuzione su un computer connesso a Internet. La richiesta viene eseguita e la risposta viene rispedita indietro al client che visualizza il documento. Il browser web *parserizza* codice HTML che e' il linguaggio usato per formattare le informazioni su web.



Il protocollo con il quale client e server comunicano sul web è il protocollo HTTP

Protocollo HTTP (HyperText Transfer Protocol)

HTTP e' un insieme di regole per scambiare informazione (testo, immagini, suono, video e altri file multimediali) su World Wide Web. E' un protocollo di livello applicativo ed e' relativo alla suite di protocolli TCP/IP, la base per lo scambio di informazioni su Internet. Il concetto essenziale di HTTP include l'idea che i file possono contenere riferimenti ad altri file selezionando i quali si avviano ulteriori richieste di trasferimento.



Un documento web viene individuato univocamente sulla rete da un indirizzo che si chiama URL

```
http://www-kdd.isti.cnr.it/index.html
```

- la prima parte indica il protocollo
- la seconda indica l'indirizzo del server web a cui ci rivolgiamo
- la terza indica il documento che richiediamo al server e che risiede sul suo disco

Una URL può essere **dinamica**, ovvero indirizzare pagine **dinamiche** che vengono costruite online e non risiedono permanentemente sul server.

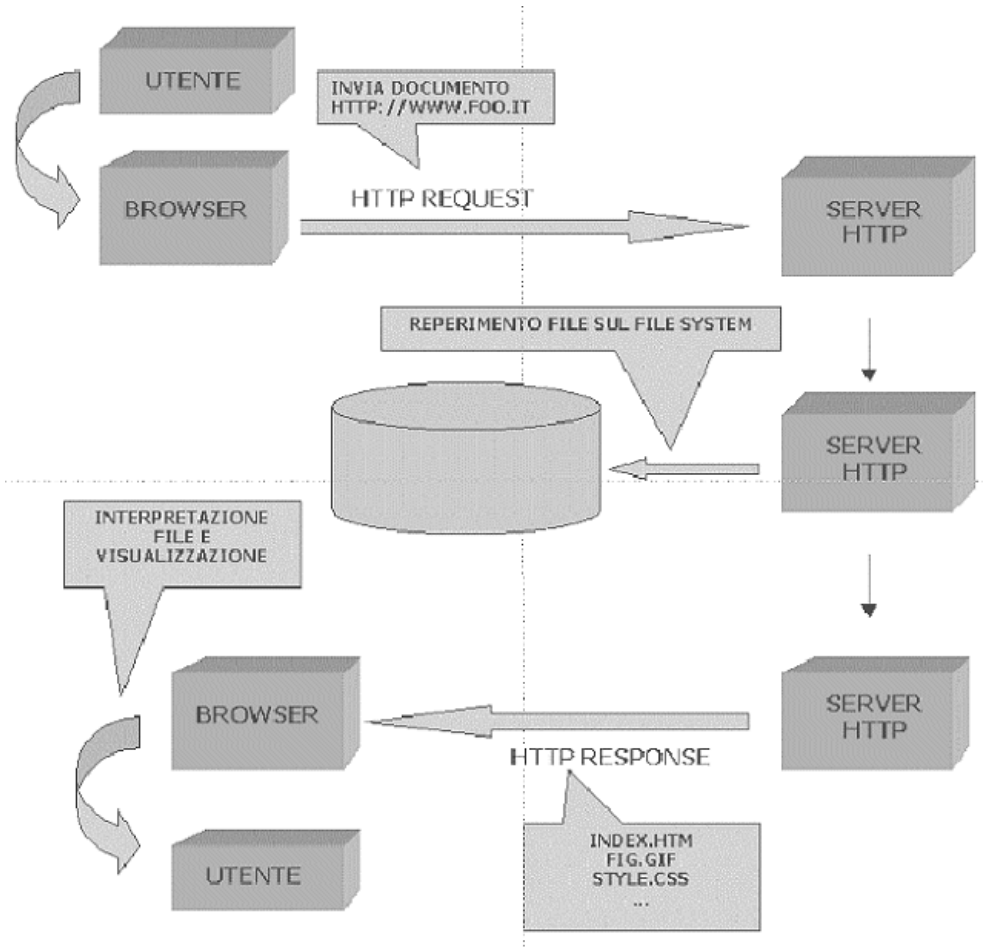
```
http://www.miodominio.it/mappa.php?name=miamappa
```

In questo caso `mappa.php` indica il programma o l'estensione del server web a cui chiediamo di eseguire un compito e le stringhe dopo il carattere `?` indicano la sequenza dei parametri passati, separati dal carattere `&`.

Ogni computer web server esegue un *HTTP server*, un programma che è progettato per aspettare richieste HTTP e gestirle quando arrivano.
Il server HTTP, come tutti i protocolli applicativi su TCP, ascolta su una *porta*.
Il meccanismo delle porte consente di scambiare dati tra processi, anche remoti, che comunicano con protocollo TCP/IP. Quando un applicativo "ascolta" su una porta significa che aspetta di ricevere dei dati leggendo da un particolare file identificato del numero della porta. Per convenzione le porte 0

- 1024 sono "riservate" ai processi di sistema, mentre le altre sono libere per processi utente. Ad esempio il servizio telnet comunica sulla porta 23, ftp sulla 21, SMTP (posta elettronica) sulla 25, HTTP sulla 80.
In pratica ogni servizio attivo su una macchina e' identificato dalla porta su cui ascolta. Il server HTTP può essere configurato per ascoltare richieste anche su altre porte, in questo modo si possono configurare più server web sulla stessa macchina.
I server web più conosciuti sono Apache (Unix,Windows) e Internet Information Server di Microsoft (IIS) che gira su piattaforma Windows.

Il browser web e' un client HTTP che manda richieste ai server HTTP. Il documento richiesto viene identificato sulla rete tramite un indirizzo univoco URI (Uniform Resource Identifier), normalmente indicato come URL (Uniform Resource Locator). Il browser si preoccupa di tradurre le URL in richieste HTTP e di parserizzare i documenti HTML per visualizzarne il contenuto.
Quando l'utente tramite un browser web richiede un documento aprendo un file web (digitando una URL) o cliccando su un hyperlink, il browser costruisce una richiesta HTTP e la manda al server indicato nella URL. Il server HTTP di destinazione riceve la richiesta e, dopo la necessaria elaborazione, il file richiesto viene spedito indietro al client.



HTTP e' un protocollo *state-less* cioè la connessione rimane aperta tra il client e il server soltanto per la durata del trasferimento del file richiesto e non viene tenuta nessuna informazione riguardo alla connessione.

I passi che vengono effettuati ad ogni richiesta ad un server HTTP sono:

1. Il client (tipicamente un web browser) inizia una connessione HTTP verso un host
2. Il server accetta la connessione
3. Il client manda una richiesta per un documento:

   GET *path from URL*

4. Il server reperisce il documento richiesto
5. Il server spedisce il documento trovato
6. Il client accetta il documento
7. Quando il trasferimento e' completo, il server chiude la connessione

8. Il client termina la connessione HTTP

Le richieste HTTP possono essere:

- GET richiesta di un documento
- POST spedizione parametri di una form
- PUT spedizione dati
- HEAD informazioni di header (*last modified*…)

Il server web restituisce al browser oltre all'eventuale documento, un codice numerico che indica l'esito dell'operazione. Ad esempio il codice *200* indica documento trovato e restituito, mentre *404* indica *document not found*, oppure *401 unauthorized.*

In un documento HTML con incorporate delle immagini la connessione viene aperta e chiusa N+1 volte dove N è il numero delle immagini riferite. In generale in un documento HTML, tutti i riferimenti a immagini generano una nuova connessione con il server.

Le immagini tipicamente possono essere "pesanti" (in Kb) e rendere lenta la navigazione di un sito.

IL SERVER WEB

Il server web e' un processo sempre attivo su una macchina connessa ad Internet, che ascolta richieste HTTP, reperisce il documento (oppure compie operationi più complesse come eseguire il programma/script) richiesto e restituisce il risultato al client web.

Il web server (o server HTTP) come tutti i servizi basati su TCP/IP, si attiva su una porta, che è il numero a cui si fa corrispondere il servizio. In questo modo una macchina può far girare più servizi differenziando le porte.
Ci sono porte standard per i vari servizi (ftp, telnet, posta elettronica, web)
La porta di default del web server è la 80, ma può essere configurato per funzionare su un'altra porta libera. In questo caso la porta va specificata nella configurazione del webserver e nella URL.

http://localhost:8080/

http://www.miodominio.it:9736/esempi/esercizio.php

Negli esempi di queste URL il web server è stato configurato per funzionare sulla porta 8080 (nel primo caso) e sulla 9736 (nel secondo caso).

Nella prima URL la parola **localhost** sta a indicare il server web attivo sulla macchina locale (che può funzionare senza una connessione internet). Quindi la macchina locale fa da server e da client.

Inoltre, nella prima URL attiviamo una richiesta al server web locale attivo sulla porta 8080 senza specificare quale documento restituirci. Il web server in questo caso decide di ritornarci il documento di *default*.
Nella seconda URL invece specifichiamo di restituirci la pagina esercizio.php nella cartella esempi.

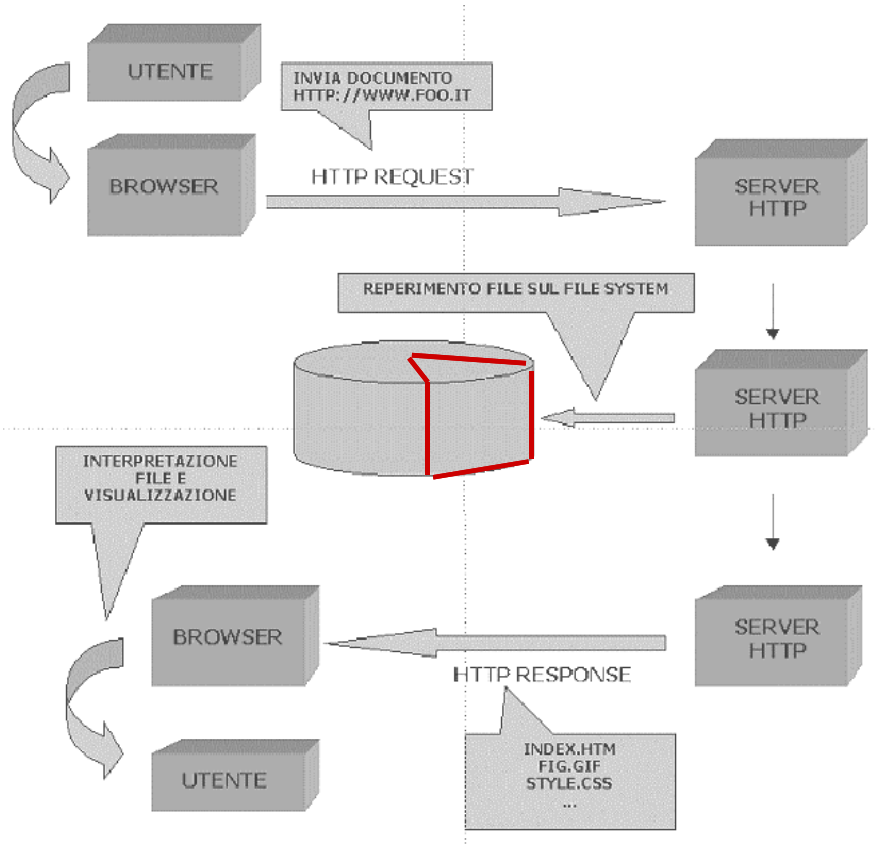
Su windows un processo sempre attivo e' chiamato *servizio* e viene gestito dall'amministratore di sistema dal pannello di controllo.
Gestire un servizio vuol dire sostanzialmente *attivarlo*, *fermarlo* e *configurarne* i parametri di funzionamento. Configurare un server web significa:

- stabilire quali directory sono visibili all'esterno;
- stabilire la porta di funzionamento;
- definire quali diritti hanno gli utenti sulle directory e/o i file in essa contenuti;

- definire il documento di default da visualizzare per ogni directory,
- ….. altre opzioni come stabilire i MIME type , virtual host etc….

Directory di default

Per ragioni di sicurezza e di pulizia del sistema, il server web ha visibilità di una sola parte del file system della macchina. La porzione di file system visibile dal server web si chiama *directory (o cartella) di default* del server web.
La cartella di default del server web varia a seconda del server considerato e del sistema operativo su cui è installato. Ad esempio per Apache su windows tipicamente è la cartella htdocs che si trova nella directory di installazione di Apache (ad es. *C:\Programmi\Apache Group\Apache\htdocs*)



http://www.miodominio.it:9736/esempi/esercizio.php

In questo caso la cartella esempi è una sotto cartella della directory di default quindi su Apache sarà in:
`C:\Programmi\Apache group\Apache\htdocs\esempi`

Affinchè il file esercizio.php sia visibile alla url di esempio sopra dovrà quindi essere salvato in questa cartella.

Alias e cartelle virtuali

Quando un server web è attivo su una macchina la sua cartella di default è automaticamente resa pubblica. Se vogliamo pubblicare un documento dobbiamo quindi copiarlo in tale cartella.

In alcuni casi può essere utile rendere pubblica anche un'altra porzione del file system. Si parla in questo caso di *alias o cartelle virtuali*
Il meccanismo sta nel definire una nuova URL che punta ad una specifica cartella del disco, non necessariamente dentro la cartella di default.

Creare un *alias* vuol dire creare una corrispondenza tra una URL e una cartella del disco

Ad esempio possiamo definire che la seguente URL:

http://localhost/esalias/

Corrisponde alla cartella:

C:\esempioalias

Apache

Il web server Apache è free opensource creato e mantenuto dall Apache foundation ed è iberamente scaricabile da **http://www.apache.org** dove si può trovare tutta la documentazione. Può essere installato sia su Windows che su Unix, anche se è stato realizzato specificatamente per Unix.

L'installazione su Windows è abbastanza agevole, può funzionare correttamente anche senza specifiche configurazioni (a parte installare l'eventuale modulo per la tecnologia serverside che si intende usare).

Per farlo partire(fermare)**:**

- Selezionare Avvia(Ferma) dal menu Apache nel menu Windows\programmi\

- Dal prompt dei comandi digitando:
  Net start(stop) apache
- Dal pannello di controllo\servizi\apache

Si può *configurare* dal file *httpd.conf*, generalmente reperibile alla voce Apache del menu programmi di windows, oppure nella cartella di installazione di Apache. E' un file di testo (editabile con blocco note) ed è suddiviso in varie sezioni:

- **Global**: parametri globali del server, si modificano solo per esigenze particolari
- **Main server**: funzionamento del server web
- **Virtual hosts**: evenutali altri host virtuali gestiti dal server

Nella sezione **Main Server** troviamo i parametri di configurazione principali:

Il parametro listen setta la porta di funzionamento, di default la 80:

```
Listen 80
```

Il nome del server viene definito dalla entry ServerName:

```
ServerName kddport.isti.cnr.it
```

La cartella di default viene definita dalla entry DocumentRoot:

```
DocumentRoot "C:/Program Files/Apache Group/Apache/htdocs“
```

Gli alias:

```
<IfModule mod_alias.c>
Alias /icons/ "C:/Program Files/Apache Group/Apache/icons/"
  <Directory "C:/Program Files/Apache Group/Apache/icons">
        Options Indexes MultiViews
        AllowOverride None
        Order allow,deny
        Allow from all
    </Directory>
```

Configurare Apache

Editare il file httpd.conf per le modifiche alla configurazione
Riavviare apache ad ogni modifica.

Nota: Apache è sensibile ai caratteri minuscoli/maiuscoli della URL!

I file di Log

Tutta l'attività di un server web viene registrata in appositi log files. Un file di log contiene informazioni per ogni richiesta arrivata al server. Tali informazioni variano in base al formato scelto (NCSA, W3C, IIS) ma tipicamente abbiamo:

- IP address del client
- Data e ora
- Nome del file richiesto
- Risposta del server
- Protocollo usato
- Tipo del browser

Esempio:

2003-03-24 16:47:29 127.0.0.1 - GET /prova.asp 200 330 HTTP/1.1 Mozilla/4.0+(compatible;+MSIE+5.5;+Windows+NT+5.0;+.NET+CLR+1.0.3705) http://localhost:8080

Per ragioni di sicurezza è sempre consigliabile controllare periodicamente i file di log per individuare eventuali tentativi di accesso non autorizzato o worm.

Esistono in rete strumenti che aiutano ad analizzare i file di log creando delle statistiche e dei filtraggi dei dati.

Se attivate apache o un qualunque server web su un PC continuamente collegato ad internet, considerare i problemi di sicurezza!

- Installare un personal firewall e tenere il server web constantemente aggiornato con opportune patches, controllare periodicamente i logs
- Generalmente IIS è più soggetto a buchi di sicurezza rispetto ad Apache
- Nessun server è totalmente sicuro, una volta che è *online*

IL LINGUAGGIO HTML[2]

I documenti visibili su web sono codificati nel linguaggio HTML che permette di formattare il testo, i collegamenti, aggiungere immagini, filmati e collegamenti ipertestuali.

HTML è l'acronimo di **Hypertext Markup Language e** permette di indicare come disporre gli elementi all'interno di una pagina: le indicazioni vengono date attraverso degli appositi marcatori, detti "tag".

Ai **TAG** viene affidata la visualizzazione e hanno differenti nomi a seconda della loro funzione. I tag vanno inseriti tra parentesi angolate (**<TAG>**), la chiusura del tag viene indicata con una "/" (è il simbolo comunemente detto "slash". Quindi: **</TAG>**). Il contenuto va inserito tra l'apertura e la chiusura del tag medesimo, secondo questa forma:

```
<TAG attributi>contenuto</TAG>
```

Ecco un esempio, con una sintassi che serve a disporre un testo giustificato a destra:

```
<P align="right">testo</P>
```

dall'esempio è evidente che la struttura di un attributo è: **attributo="valore"**

Quindi in definitiva la struttura di un tag sarà:

```
<TAG attributo_1="valore1" attributo_2="valore2">contenuto</TAG>
```

Alcuni particolari tag non hanno contenuto - perché ad esempio indicano la posizione di alcuni elementi (come il tag delle immagini) -, conseguentemente questi tag non hanno neanche la chiusura. La loro forma sarà dunque:

```
<TAG attributi>
```

Ecco un esempio di immagine:

```
<IMG width="20" height="20" SRC="miaImmagine.gif" ALT="alt">
```

come si vede il tag non viene chiuso. Questo tipo di tag viene detto "empty", cioè "vuoto".

[2] Tratto in parte da www.html.it

Annidamento e indentazione

Una caratteristica importante del codice HTML è che i tag possono essere annidati l'uno dentro l'altro. Anzi, molto spesso è necessario farlo.

Ad esempio:

```
<TAG1 attributi>
    contenuto 1

    <TAG2>
        contenuto 2
    </TAG2>

</TAG1>
```

Un documento HTML è normalmente diviso in due sezioni:

Testa (<head>):
: Contiene informazioni non immediatamente percepibili, ma che riguardano il modo in cui il documento deve essere letto e interpretato. Questo è il luogo dove scrivere - ad esempio - i meta-tag (alcuni sono ad esclusivo beneficio dei motori di ricerca), script JavaScript, fogli di stile, eccetera

Corpo (<body>)
: Qui è racchiuso il contenuto vero e proprio del documento

```
<html>
<head>
  <title>La mia Pagina</title>
</head>
<body>

  <!-- Scriveremo qui -->

  Qui il nostro contenuto

</body>
</html>
```

Colori

I colori vengono indicati in HTML con la notazione esadecimale. Ad esempio per definire il colore di sfondo di una pagina:

```
<body bgcolor="#0000FF">
```

Analogamente possiamo cambiare il colore dei font:

```
<font color=”#FF00FF”>
```

Ecco una tabella con la notazione di alcuni colori (molti di essi sono disponibili anche nelle varianti "dark" e "light", ad esempio: "darkblue", "lightblue"):

colore	parola chiave	Notazione esadecimale
arancione	orange	#FFA500
blu	blue	#0000FF
bianco	white	#FFFFFF
giallo	yellow	#FFFF00
grigio	gray	#808080
marrone	brown	#A52A2A
nero	black	#000000
rosso	red	#FF0000
verde	green	#008000
viola	violet	#EE82EE

I colori visibili correttamente su Browser web sono detti *browser-safe* e sono 256. I colori principali sopra elencati sono tutti browser-safe. Esistono tabelle di tutti i colori browser safe, facilmente referibili sul web, ad es: http://www.visbone.com/colorlab

Per centrare una frase possiamo usare il tag .

```
<center> questa frase è centrata </center>
```

I caratteri possono essere resi **grassetto** con il tag <b>, *corsivo* con il tag <i> o , o sottolineato con <u>. Per una guida esaustiva di tutti i tag di formattazione riferirsi ad un manuale HTML (ad esempio su www.html.it).

Il tag <P> aggiunge un nuovo paragrafo mentre il carattere
 corrisponde ad andare a capo.

Link

Il link è un collegamento, un ponte tra una pagina è l’altra.
Non tutti però sanno che i link testuali hanno diversi stati:

Status	Codifica in HTML 4.01	Descrizione
Collegamento normale	link	Normalmente il link quando si trova "a riposo" viene evidenziato in qualche maniera all'interno della pagina HTML, in modo che sia facile per l'utente individuarlo. Nell'HTML tradizionale il link è sempre sottolineato (è possibile eliminare la sottolineatura soltanto usando i CSS). Di default i link sono blu (**#0000FF**).
Collegamento visitato	visited	Un link è visitato, quando l'URL della pagina compare nella cronologia dell'utente. Di default i link visitati sono di color violetto (più esattamente: **#800080**).
Collegamento attivo	active	Il collegamento è attivo nel momento in cui il link è stato cliccato e sta avvenendo il passaggio da una pagina all'altra.

```
<body link="red" alink="yellow" vlink="green">
```

Di solito per spiegare che cosa sono i link si utilizza la metafora dell'ancora con "la testa" all'interno del documento stesso, e la "coda" in un altro documento (o all'interno di un altro punto del documento stesso).

Link che puntano ad altri documenti

Ecco la sintassi per creare un link con riferimento a un sito web:

```
Le risorse per webmaster sono su <a href="http://www.html.it/">
HTML.IT </a>.
```

Che dà come risultato: 'Le risorse per webmaster sono su HTML.IT'.

Come si può intuire la testa della nostra ancora è il testo "HTML.IT", mentre la coda, cioè la destinazione (specificata dall'attributo **href**) è il sito web verso cui il link punta, cioè http://www.html.it.

È indifferente che la destinazione dell'ancora sia una pagina HTML di un sito, un'immagine, un file pdf , un file zip, o un file exe: il meccanismo del link funziona allo stesso modo indipendentemente dal tipo di risorsa; poi il browser si comporterà in modo differente a seconda della risorsa.

Ad esempio:

- Immagine .gif, .jpg, .png: Viene visualizzata nel browser

- Documento .html, .pdf, .doc: La pagina è visualizzata nel browser.

Nel caso dei documenti .doc e .pdf l'utente deve avere installato sul proprio pc l'apposito plugin (nella maggior parte dei casi è sufficiente che abbia installato rispettivamente Microsoft Word e Adobe Acrobat Reader). Se non è installato il plugin il sistema chiederà all'utente se salvare il file.

- File .zip, file .exe: Viene chiesto all'utente di scaricare il file

NOTA bene: per motivi di sicurezza non è possibile eseguire un file ".exe" direttamente dal web; l'utente dovrà sempre prima scaricarlo sul proprio PC.

Potete anche specificare un indirizzo email. In questo caso si aprirà direttamente il client di posta dell'utente con l'indirizzo e-mail pre-impostato. La sintassi è la seguente:

```
<a href="mailto:tuaMail@nomeTuoSito.it">Mandami un'e-mail</a>
```

Che dà come risultato: Mandami un'e-mail

Naturalmente possiamo inserire anche delle immagini.

I formati ammessi nel Web sono sostanzialmente tre:

- **GIF** (**Graphic Interchange Format**). Le GIF sono immagini con non più di 256 colori (dunque con colori piatti e senza sfumature), come grafici o icone

- **JPG**: è l'acronimo del gruppo di ricerca che ha ideato questo formato (il **Joint Photographic Experts Group**), idoneo per le immagini di qualità fotografica

- **PNG** (**Portable Network Graphic**). Il PNG è un tipo di immagine introdotto abbastanza di recente, elaborato dal W3C per risolvere i problemi di copyright del formato GIF (che è appunto proprietario); oggi il PNG è letto oramai da tutti i browser e offre alcune caratteristiche che gli altri formati non hanno.

Inoltre è importante ricordare che il codice HTML fornisce delle indicazioni al browser su come visualizzare il testo e le immagini - ed eventualmente i video e i suoni - all'interno della pagina.
Il testo (come abbiamo visto) è scritto direttamente nel file HTML, le immagini invece sono caricate insieme alla pagina. Attenzione dunque a non inserire immagini troppo pesanti (ricordatevi di ottimizzare sempre i file); bisogna

evitare inoltre di sovraccaricare la pagina con troppe immagini. Allo stato attuale dell'arte infatti la maggior parte degli utenti (e non soltanto quelli italiani) naviga ancora con un modem analogico da 56 Kbs: inserire troppe immagini significa dunque creare pagine lente da caricare. Per ottenere un sito web dalla grafica accattivante, spesso è sufficiente giocare con i colori dello sfondo e delle scritte.
La sintassi per inserire un'immagine è:

```
<img src="miaImmagine.gif">
```

dove:

- **img** significa **image**, cioè **immagine**
- **src** significa **source**, cioè origine

Il tag è un tag "vuoto", che non ha la necessità di essere chiuso.
Ecco ad esempio come inserire il logo di HTML.it in una pagina dallo sfondo blu (si presuppone che il logo si trovi nella stessa cartella del file HTML):

```
<img src="logo.gif">
```

Un'immagine può essere un link ad una pagina o una nuova immagine, circondando il tag IMG con l'anchor:

```
<a href="dettagliimmagine.html"><img src="miaImmagine.gif"></a>
```

Tabelle

Le tabelle sono una delle parti più importanti di tutto il codice HTML: nate sin dagli inizi del Web per impaginare dati aggregati, si sono poi trasformate in uno strumento indispensabile per gestire i layout grafici.

Il loro ampio utilizzo all'interno dei documenti ha fatto sì che – nel passaggio dall'HTML 3.2 all'HTML 4 - le specifiche delle tabelle venissero estese con una serie di notazioni destinate a "far ordine" all'interno di un codice che rischiava di diventare troppo vasto.
Immaginiamo la nostra prima tabella come una griglia formata da righe e colonne. I tag necessari per creare una tabella sono:

<table> apre la tabella
<tr> "table row": indica l'apertura di una riga

<td> “table data”: indica una cella all’interno di una riga

Ecco un primo esempio di tabella:

```
<table border="1>
    <tr>
       <td>prima cella</td>
       <td>seconda cella</td>
   </tr>

   <tr>
      <td>terza cella</td>
      <td>quarta cella</td>
   </tr>
</table>
```

Che viene visualizzato così:

prima cella seconda cella
terza cella quarta cella

FORM in HTML

I moduli (o form) sono il modo per interagire con l’utente. Tramite i moduli, l’utente puo’ inserire dati ed esprimere preferenze. Tramite le FORM si può infatti realizzare una spedizione di dati dal *browser al server*. Il server li riceve e poi deciderà il da farsi. Il trattamento dei dati provenienti da moduli rientra nell’ambito delle tecnologia server-side che vedremo più avanti.

<FORM></FORM>

Questo tag apre e chiude il modulo e raccoglie il contenuto dello stesso, che puo' variare a seconda dei tag che vedremo oltre. Non e' possibile inserire un modulo all'interno di un altro. In altre parole i form non permettono nidificazioni.

La sintassi usuale dei tag analizzati e' la seguente:

<FORM method="get|post" action="http://www.tuosito.com/cgi-bin/nome_script.cgi">

Se method e' impostato come GET i dati vengono spediti al server e separati in due variabili. **Usando questo metodo i parametri sono visibili nella barra dell’indirizzo del browser**. Per questo metodo il numero massimo di caratteri contenuti nel form e' di 255.
Utilizzando "method=POST" i dati viaggiano come messaggio HTTP e non sono visibili.

Questa caratteristica fa si' che lo script possa leggere una quantita' illimitata di caratteri.
La *action* della form indica quale componente del server web (dipende dalla tecnologia server-side utilizzata) tratterà i dati ricevuti.

Impostato il primo tag <FORM> del modulo e' possibile, sempre che la componente lato server lo permetta, creare alcuni elementi molto utili al fine di una corretta gestione dei dati.

Il tag di base per la definizione degli elementi di un form e' <INPUT>, che viene utilizzato per aggiungere pulsanti, menu di scelta, password ecc. a <INPUT> possono venire assegnati 8 valori differenti, che andiamo a spiegare.

<INPUT type="TEXT" name="nome" maxlength="40" size="33" value="Il tuo nome">

Questo valore crea i tipici campi di testo, dove usualmente vengono richiesti dati quali il nome o l'indirizzo e-mail. E' un valore usato soprattutto per informazioni non predefinite che variano di volta in volta. TEXT ha tre attributi aggiuntivi presenti anche nell'esempio: **maxlength** (il numero massimo di caratteri inseribili nel campo, oltre il quale non e' possibile aggiungere), **size** (la larghezza della stringa all'interno della pagina) e **value** (visualizza un testo di default all'interno della stringa).

ESEMPIO Il tuo nome

<INPUT type="PASSWORD" name="nome" maxlength="40" size="33">

Questo campo funziona similmente a TEXT visto in precedenza, ma con una piccola differenza che ne giustifica il nome: quando si digita all'interno della stringa bianca, non appaiono le lettere ma i classici asterischi delle password. In realta' i dati non vengono in alcun modo codificati, per cui rimane un'insicurezza di fondo che questo comando non elimina.

ESEMPIO

<INPUT type="CHECKBOX" name="età" value="yes" checked>

Questo attributo viene solitamente utilizzato per informazioni del tipo "si/no" e "vero/falso". Crea delle piccole caselle quadrate da spuntare o da lasciare in bianco. Se la casella e' spuntata, input restituisce un valore al CGI, al contrario non restituisce alcun valore. **Value** impostato su **"yes"** significa che di default la casella e' spuntata. **Checked** controlla lo stato iniziale della casella, all'atto del caricamento della pagina.

ESEMPIO

<INPUT type="RADIO" name="giudizio" value="sufficiente">

<INPUT type="RADIO" name="giudizio" value="buono">

<INPUT type="RADIO" name="giudizio" value="ottimo">

Questo attributo ha funzioni simili a quello visto in precedenza, ma presenta piu' scelte possibili. Selezionando una voce tra quelle presenti, qualora abbiano tutte valore "name" identico, si deselezionano automaticamente le altre.

SUFFICIENTE
BUONO
OTTIMO

<INPUT type="SUBMIT" value="spedisci">

Il classico bottone che invia il form con tutti i suoi contenuti. La grandezza del bottone dipende dalla lunghezza del testo.

ESEMPIO Spedisci

<INPUT type="RESET" value="Ricomincia">

Bottone che reimposta l'intero form eliminando i dati inseriti.

ESEMPIO Ricomincia

<INPUT type="IMAGE" src="pulsante.gif">

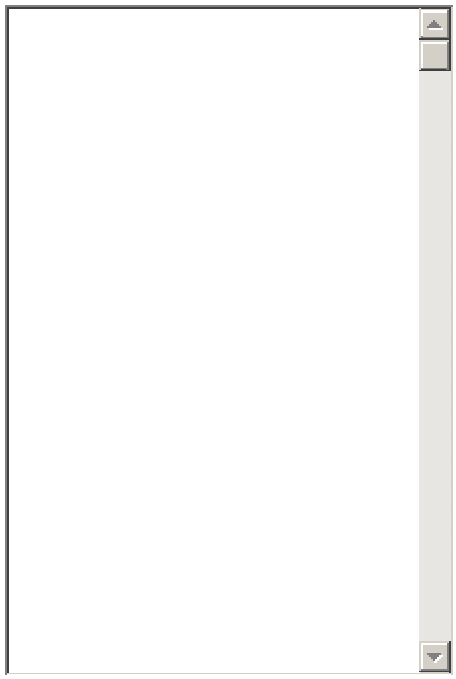
Funzione simile a quella del tasto "SUBMIT" ma con la differenza che al posto del bottone di default, viene visualizzata un'immagine.

ESEMPIO

<TEXTAREA cols=40 rows=5 WRAP="physical" name="commento"></textarea>

Textarea e' utilizzato per commenti o campi che prevedono l'inserimento di molto testo. La larghezza e' impostata da **"cols"** e l'altezza da **"rows"**. **WRAP="physical"** stabilisce che qualora il testo inserito superi la larghezza della finestra, venga automaticamente riportato a capo.

ESEMPIO



<SELECT size=1 cols=4 NAME="giudizio">

<OPTION selected Value=nessuna>

<OPTION value=buono> Buono

<OPTION value=sufficiente> Sufficiente

<OPTION Value=ottimo> Ottimo

</select>

Select e' un elemento che permette la creazione di elenchi a discesa con varie possibilità di scelta. Nel nostro esempio abbiamo simulato la richiesta di un giudizio su un sito Web.

ESEMPIO 

Il campo di tipo HIDDEN è un campo non visibile nel browser, ma serve a conservare valori da passare come parametri

<INPUT TYPE=HIDDEN NAME=MAILFORM_SUBJECT VALUE="Titolo del form">

in questo caso sarà possibile trasmettere al server il valore “titolo del form” senza che questo sia visibile all’utente.

Esempio di FORM

```
<FORM ACTION=”gestisciform.cgi” METHOD=POST>

<B>Nome e cognome</B><BR>

<input type=text NAME=MAILFORM_NAME size=33><BR><BR>

<B>Indirizzo E-mail</B><BR>

<input type=text NAME=MAILFORM_FROM size=33><BR><BR>

<B>Commenti</B><BR>
<TEXTAREA NAME=MAILFORM_TEXT ROWS=10 COLS=42 WRAP>
</TEXTAREA><P>

<INPUT TYPE=SUBMIT VALUE="Spedisci"> <INPUT TYPE=RESET
VALUE="Cancella">

</FORM>
```

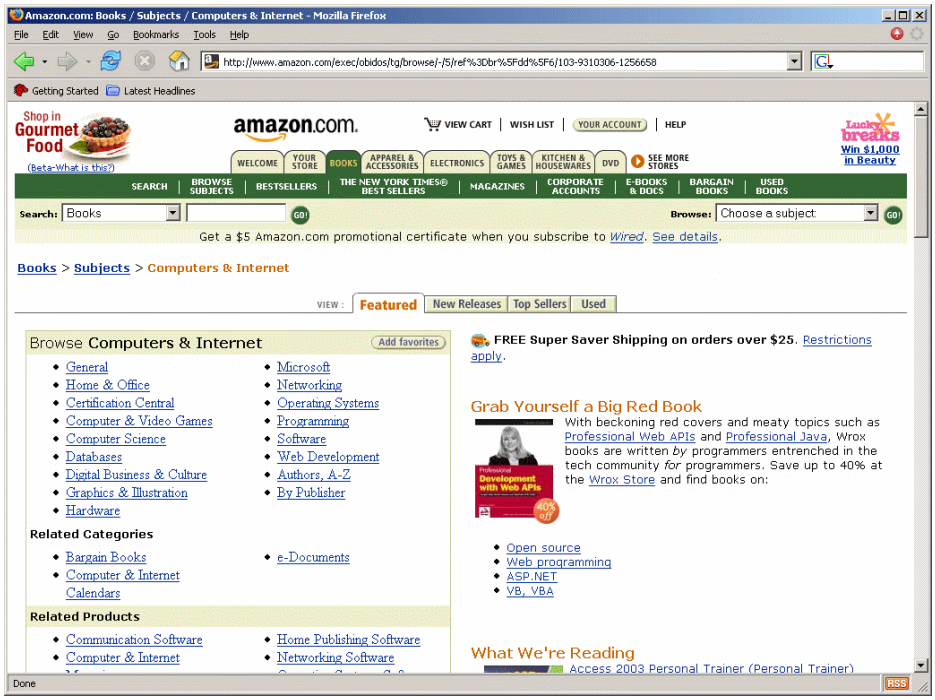
Considerazioni

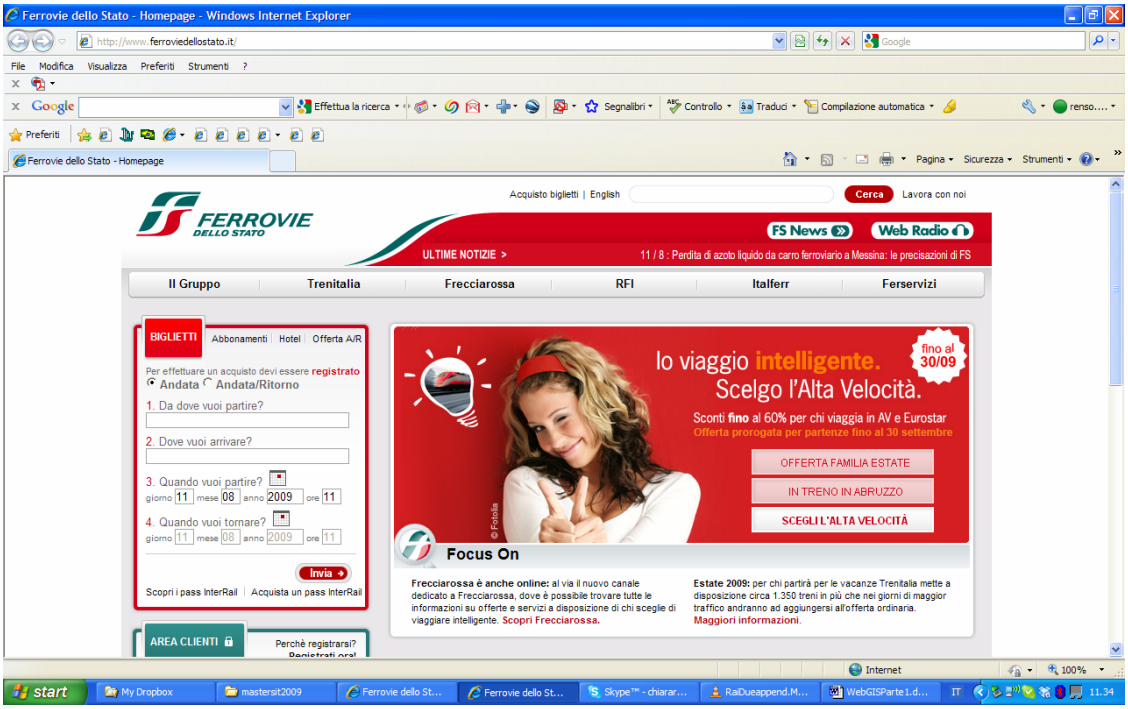
Per dettagli su tutti i TAG HTML vedere riferimenti su web.
In aggiunta all’HTML, da alcuni anni si adottano fogli di stile (CSS, cascading style sheet) che permettono di separare il contenuto dalla presentazione. Inoltre, la versione di HTML di cui è fortemente consigliato l’uso è XHTML.
Per realizzare un sito web non basta conoscere i TAG HTML o usare un tool di editing avanzato (MS Frontpage o Macromedia Dreaweaver). Occorre usare tutta una serie di accorgimenti per rendere il sito graficamente piacevole, usabile, accessibile. Questo settore prende il nome di Web Design.

TECNOLOGIE WEB

L'infrastruttura web si basa sul protocollo HTTP. Ma questo non permette lo sviluppo di *applicazioni* complesse che coinvolgano una fase di elaborazione oltre che di passaggio di dati.
In una applicazione web l'utente interagisce con il sito, immettendo dati o preferenze e avendo come risposta pagine personalizzate. Le applicazioni web tipicamente offrono dei *servizi all'utente*, mentre un sito "statico" o "vetrina" offre una sola visione delle pagine. Inoltre le applicazioni web offrono la possibilità di trattare grandi quantità di pagine (pagine costruite dinamicamente).

Alcuni esempi classici:





Applicazioni complesse come quelle di WebGIS necessitano di un elevato grado di processing (ricerche, analisi, mapping…) che il protocollo http non supporta.

Dove eseguiamo questa computazione? Con quali tecnologie? Solo dalla parte server? O è meglio spostare parte del peso della computazione anche sul client? E se si, quanto? Quali sono i vantaggi di entrambi gli approcci?

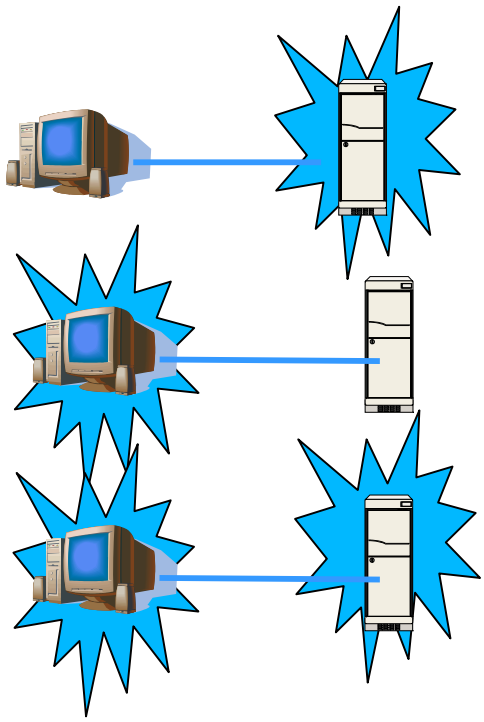
Il modello Client-Server permette la condivisione di informazioni e il livello di *processing* e' modificabile in base ad un certo numero di fattori che determinano tale scelta (mercato, esperienza dell'utente, connessione Internet, potenza di calcolo del computer…).
Per questo motivo sono state sviluppate tecnologie per permettere una maggiore interazione dell'utente con il server web e una capacità di elaborazione sia del server che del client web.

I programmi eseguiti dal server web sono detti *server-side*, mentre le tecnologie che aggiungono potere di calcolo al client sono dette *client-side*.

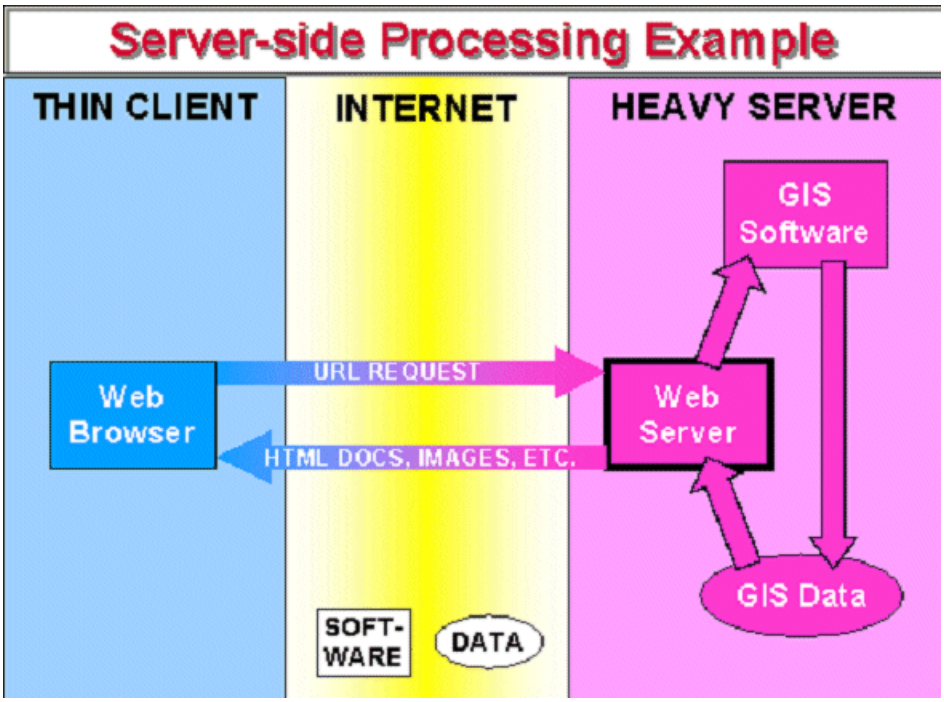
tecnologia server side:
peso della computazione risiede tutta sul server

tecnologia client-side
la computazione avviene principalmente sul browser.

Generalmente le applicazioni web complesse usano strategia ibrida



Strategia Lato Server



I **vantaggi** di una tecnologia server side sono quelli di avere un sito centralizzato e quindi:

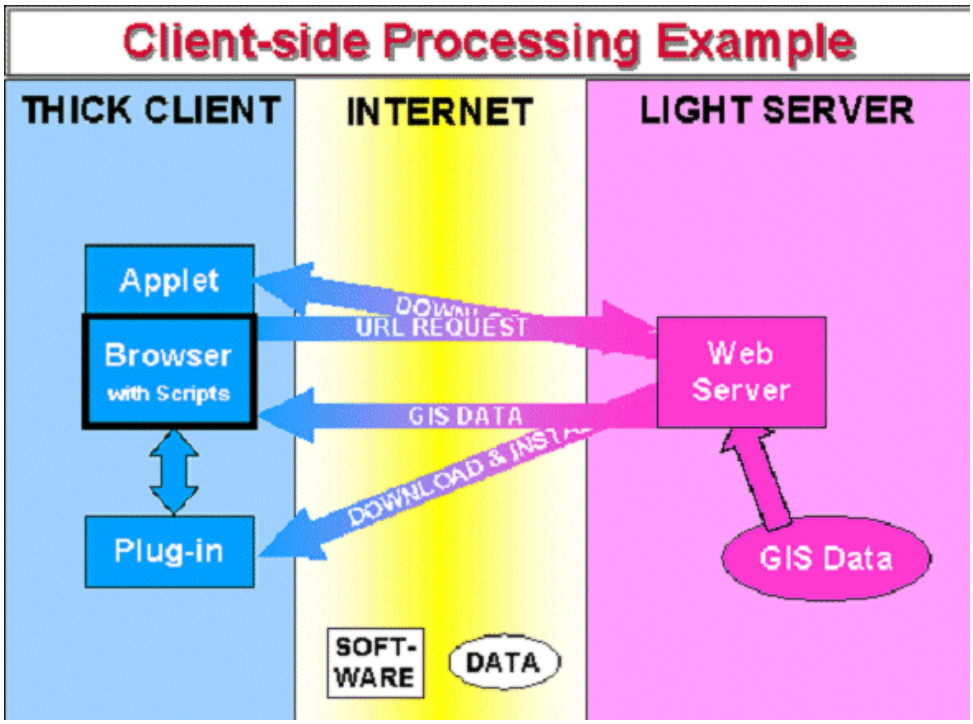
- più facile da mantenere e tenere aggiornato
- più semplice da controllare e gestire gli accessi
- avere un server potente permette:
- funzionalità di processing avanzate
- accesso a datasets grandi e complessi che non sarebbero trasferibili via Internet

Gli **svantaggi** del server side si hanno quando il processore deve eseguire compiti particolarmente pesanti e quando si devono trasferire sulla rete grossi volumi di dati, e quindi:

- il tempo di risposta può crescere considerevolmente
- la capacità di calcolo del client non viene sfruttata
- la comunicazione via Internet e la computazione da parte del server è necessaria per ogni richiesta, il che accresce il traffico della rete e il carico del server

Strategia Lato Client

Usare una tecnologia client side significa spostare parte o tutto il peso della computazione sulla parte client (browser).



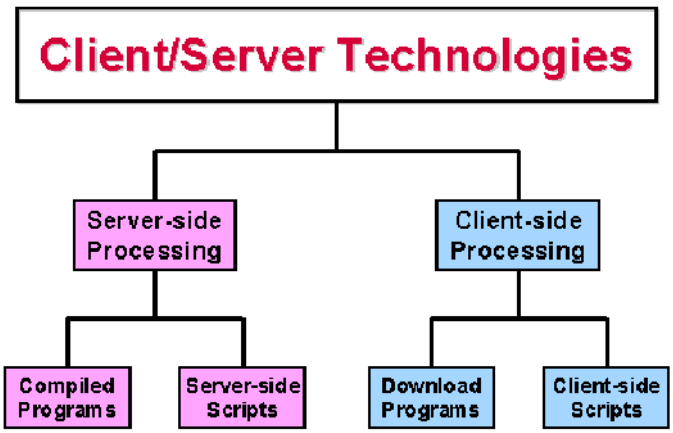
I browser web sono chiamati "thin client" cioè client che non hanno potere di calcolo, ma solo capacità di visualizzazione. Con tecnologia lato client i browser web diventano "thick" cioè includono altre tecnologie come Java Applets, Active X e Plug-in.

I **vantaggi** di usare una tecnologia client side stanno nel fatto che i browser diventano "thick clients", e quindi:

- aiutano a superare gli svantaggi del server-side (riducono il carico del server e decrementano il traffico in rete)
- danno maggiore autonomia all'utente ad esempio per map browsing (pan, zoom), controllo della visualizzazione dei layers, input di query spaziali...
- permettono il trasferimento di dati in forma vettoriale (più piccoli, più veloci, più versatili)

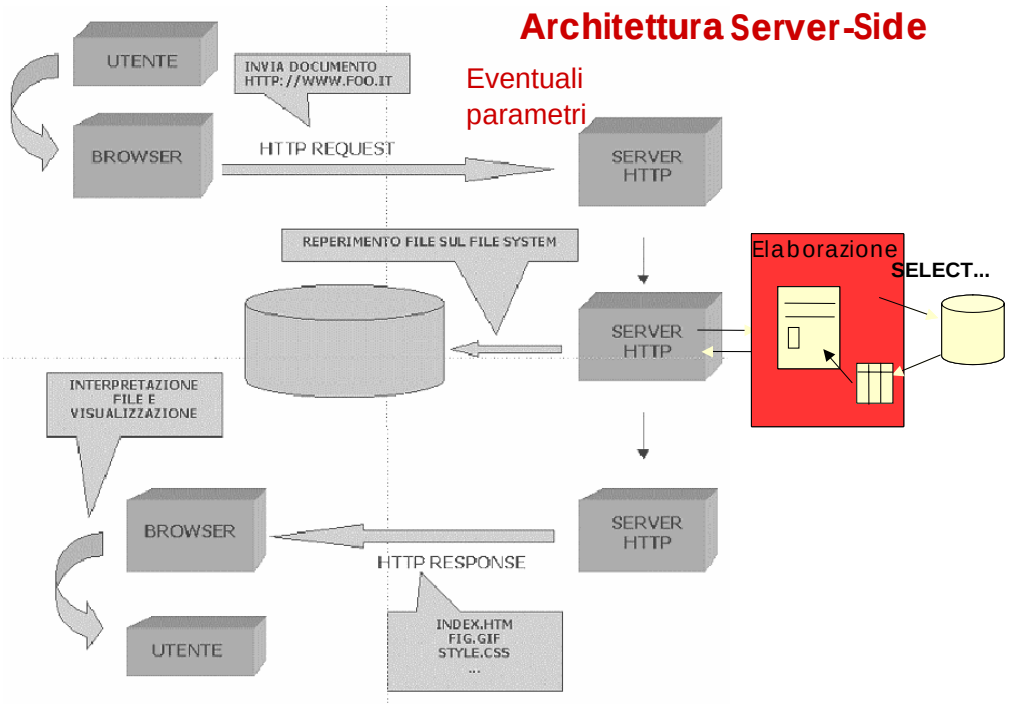
Gli **svantaggi** del client-side sono:

- downloading di applet
  - la dimensione del file è proporzionale alle funzionalità
  - l'utente può non essere disposto ad aspettare
- downloading e installazione di plug-in
  - come per gli applet
  - inoltre, tempo e sforzo extra per l'installazione e il mantenimento (aggiornamenti etc)
- dowloading iniziale di datasets anche grandi
- il computer client può essere "weak" cioè non avere grossa potenza di calcolo



Tecnologia Web Server-Side

Con tecnologia web *server-side* si indica un insieme di meccanismi che permettono al server web di elaborare informazione. Il server web non si limita solo a rispondere a richieste HTTP restituendo documenti HTML, ma e' in grado di eseguire anche una fase di elaborazione dei dati.
L'utente può quindi interagire con il server ad esempio sottomettendo dati che il server elabora e restituisce poi la risposta sotto forma di pagina HTML. Un caso tipico è l'accesso da parte dell'utente ad un database che risiede sul server. In questo caso la pagina web funziona come una interfaccia per accedere ai dati che risiedono sul server.
Si realizza in questo modo un meccanismo di interazione tra l'utente web e le applicazioni/dati che risiedono su un server centralizzato.

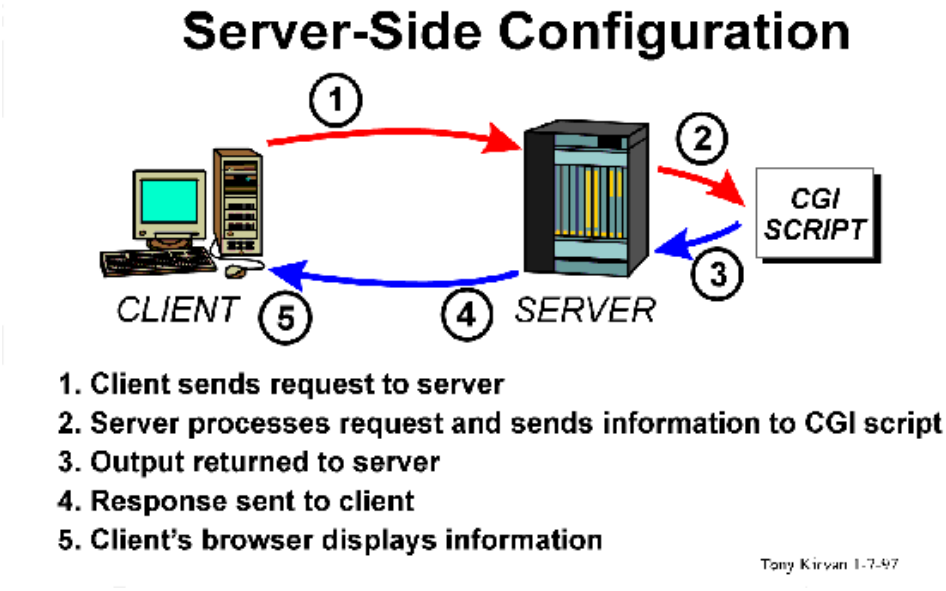


Ci sono varie tecnologie server side utilizzate tra cui Common Gateway Interface (CGI), ASP, PHP, JSP e Java Servlet.

Common Gateway Interface (CGI)

CGI (Common Gateway Interface) permette di costruire pagine *dinamiche*, cioe' pagine che non risiedono staticamente sul server, ma che vengono *"costruite" dinamicamente* da un programma CGI in dipendenza di dati che

prevengono da altre fonti, ad esempio inseriti dall'utente o che risiedono su un database esterno.
Il browser puo' richiedere di eseguire un programma CGI sul server. I CGI sono particolari programmi (eseguibili o script) che vengono eseguiti sulla macchina server e che ritornano l'output al browser.



Un tipico esempio di uso di programmi CGI e' il trattamento delle *form* HTML. Le Form in HTML sono un meccanismo per permettere all'utente di immettere dati e di attivare in conseguenza dell'invio dei dati, una applicazione sul server (la ACTION della Form).

Quando un utente compila una form su una pagina web e la sottomette, generalmente c'e' bisogno di una qualche elaborazione da un programma applicativo. Il server web passa le informazioni della form ad un programma che ne elabora i dati e poi spedisce indietro al browser una risposta. Questo meccanismo fa parte del protocollo HTTP.

Esempio di FORM in HTML:

```
<HTML>
.......
<FORM name=prova action="helloworld.pl"
METHOD=GET>
 <INPUT TYPE=text NAME="Nome">
 <INPUT TYPE=submit VALUE="submit" >
</FORM>
....
</HTML>
```

Il passaggio dei dati tra una form HTML e il CGI puo' avvenire secondo due metodi: GET e POST. Passare i dati di una form con metodo GET significa codificare i dati nella URL. Esempio di URL con passaggio di dati GET:

http://146.48.82.93/webgis/helloworld.pl?Nome=Chiara

Con il metodo POST invece i dati sono passati con un messaggio di tipo POST, come definito nello standard HTTP e non appaiono nella URL.

Alcuni tra i linguaggi più diffusi per scrivere applicazioni CGI sono C, C++, Java e Perl

Esempio CGI "Hello World"

Pagina HTML visualizzata sul browser:

Sorgente HTML

```
<HTML>
<HEAD>
<TITLE>Hello World </TITLE>
</HEAD>
<BODY>
<H2> Hello World </H2>

<p><font face="Arial,Helvetica">

<FORM name=prova action="helloworld.pl"
METHOD=GET>

Inserire il proprio nome: <INPUT TYPE=text
NAME="Nome">


<P>
Premere Submit per attivare il CGI Hello World
<P>
<INPUT TYPE=submit VALUE="submit" >


</FORM>
</BODY>
</HTML>
```

Nella action della form si indica quale programma viene mandato in esecuzione dal server quando l'utente sottomette di dati.

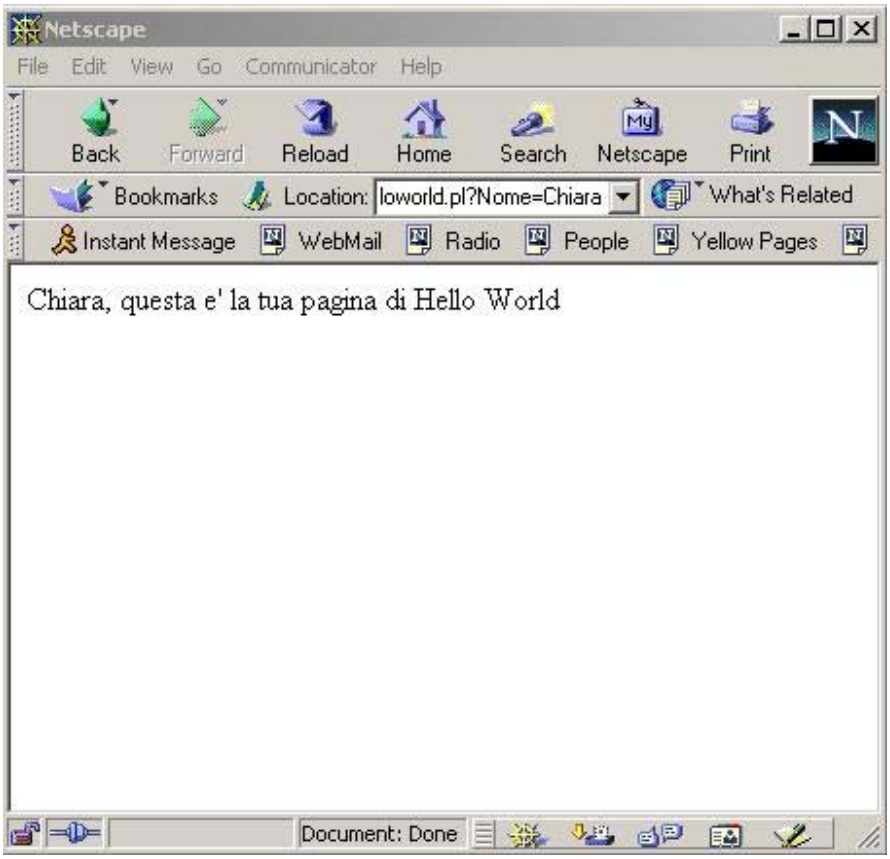
Qui, ad esempio, il server manda in esecuzione il programma helloword.pl che risiede nella directory di default del server web.

Questo e' un programma molto semplice che si limita a ricevere in input dati da parte dell'utente e restituirli al browser. In generale in applicazioni web più complesse i dati inseriti vengono poi elaborati dal server, ad esempio inserendoli in una base di dati, oppure inviandoli per mail, oppure passandoli ad un'altra applicazione che può risiedere sul server stesso o in rete. In linea teorica, il CGI può comunicare con qualunque applicazione

con cui si possa interfacciare il linguaggio con cui e' scritto il CGI.

Pagina Risultante

Supponiamo che il nome inserito sia "Chiara", otteniamo:



Notiamo come nella casella di location appaia il parametro passato con il metodo GET. Se avessimo usato il metodo POST il parametro non sarebbe stato visibile nella URL

Esempio di sorgente del programma helloword.pl scritto in Perl

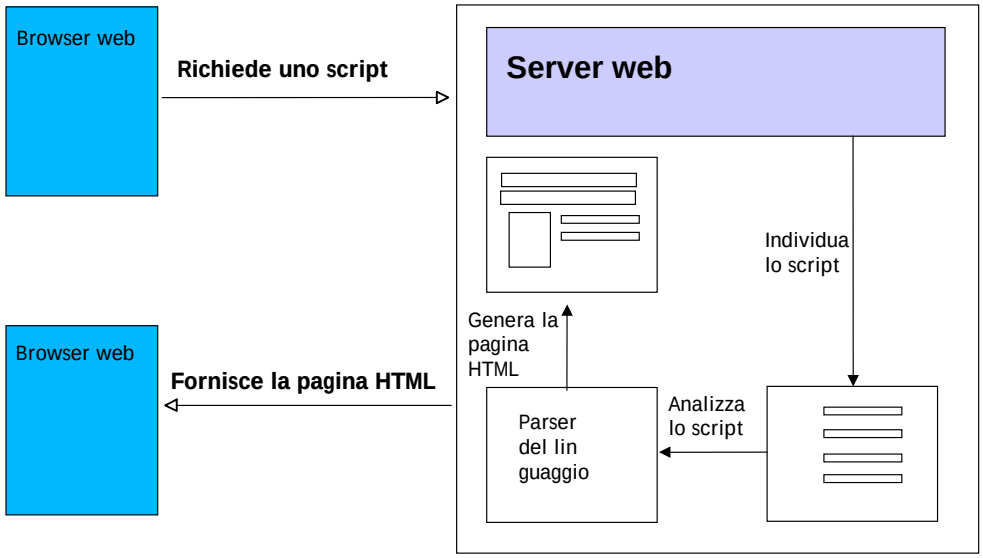
```
#!usr/local/bin/perl

    local(%in) ;
    local($name, $value) ;

    # Resolve and unencode name/value pairs into %in
    foreach (split('&', $ENV{'QUERY_STRING'})) {
        s/\+/ /g ;
        ($name, $value)= split('=', $_, 2) ;
        $name=~ s/%(..)/chr(hex($1))/ge ;
        $value=~ s/%(..)/chr(hex($1))/ge ;
        $in{$name}.= "\0" if defined($in{$name}) ;
    # concatenate multiple vars
        $in{$name}.= $value ;
    }
    print "Content-type: text/html\n\n";
    print $in{$name} ;
    print ", questa e' la tua pagina di Hello World";
```

Linguaggi di scripting

Nelle tecnologie serverside con linguaggi di scripting il server web (eventualmente equipaggiato con dei moduli aggiuntivi) riesce a individuare la parte di script della pagina, analizzarlo, eseguirlo e generare la pagina HTML risultante da rispedire al browser



Active Server Pages

Una tecnologia server-side alternativa ai CGI e' stata introdotta da Microsoft con Active Server Pages (ASP). ASP è un linguaggio di scripting, cioè permette di scrivere pagine HTML *dinamiche.* Le pagine HTML sono intervallate da script in linguaggio VBScript o Javascript che viene elaborato dal server web prima di restituire la pagina risultante al browser.

ASP e' supportato principalmente dal server web di Microsoft (IIS). Una delle caratteristiche più importanti di ASP è la capacità di istanziare ed usare componenti programmabili. Queste componenti possono essere create con tool quali Visual Basic, Visual C++, Visual J++ e altri ambienti di sviluppo. Questo meccanismo permette di integrare applicazioni web con sistemi client/server esistenti. Attualmente la tecnologia ASP si è evoluta in ASP.NET che in realtà è una tecnologia ibrida.

Esempio di FORM submit

Pagina di *Hello World*, nelle versioni con metodo GET e metodo POST. Notiamo come nella casella URL del browser in un caso (GET) appaia la stringa immessa, mentre nell'altro (POST) non appare.

Sorgente della pagina *helloword.html* per metodo GET:

```
<HTML>
<HEAD>
  <TITLE> Hello World </TITLE>

</HEAD>
<BODY>
<H2> Hello World</H2>

<P>
<H3> Esempio ASP metodo GET </H3>

<form name="prova" action="result.asp"  METHOD=GET>
Inserire il nome: <input TYPE=text  NAME="textstring">
<P>
Premere Submit per spedire i dati
<P>
<input TYPE=submit VALUE="submit" >

</form>
</BODY>
</HTML>
```

All'evento Submit della form il server HTTP attiva la pagina **result.asp** il cui codice VBScript viene eseguito dal server prima di rimandare al browser il risultato.

Sorgente di result.asp per il metodo GET

```
<% nomeinserito = request.querystring("textstring") %>

<HTML>
<HEAD>
    <TITLE> Hello World con ASP </TITLE>
</HEAD>
<BODY>

<H1> Hello World </H1>

<%=nomeinserito%>, questa e' la tua pagina di Hello World

</BODY>
</HTML>
```

Il codice ASP in una pagina HTML e' sempre delimitato dai caratteri **<%** e **%>** che individuano il codice eseguito dal server web prima di rispedire il documento al browser. Ricordiamo che al browser torna sempre un documento HTML.
Attualmente ASP non è più supportato da microsoft perché sostituito dal framework .NET (ASP.NET)

Cold Fusion

http://www.adobe.com
Il pacchetto ColdFusion offre quattro componenti:
ColdFusion Markup Language - usato assieme all'HTML
ColdFusion Studio - un editor e un tool per sviluppare applicazioni
ColdFusion Application Server - legge HTML, CFML e produce HTML
ColdFusion Administrator - per amministrare l'application server.

Server Side processing:

si integra con il web server (Apache, IIS...) In ogni momento è in esecuzione una sola istanza di CF Application server. Le pagine CF sono sul disco assieme

alle pagine HTML standard. Il server legge una pagina CF ed esegue tutti i tag CF, generando una pagina HTML che viene restituita al browser.
Il CFML fornisce il contenuto delle pagine, mentre l'HTML definisce il layout.
I file con tag CF hanno l'estensione .cfm che il server web riconosce, interpreta e restituisce il risultato al browser.

```
<CFOUTPUT>
        Hello #name#, Welcome to my website
</CFOUTPUT>
```

Il vantaggio di CF è la facilità d'uso e l'interoperabilità con vari web server. Inoltre ha primitive nel linguaggio le connessioni con il database, le query e la manipolazione dei records.

```
<CFQUERY NAME="queryname"
        DATASOURCE="dsname"
        DBTYPE="ODBC">
   query SQL...
</CFQUERY>
```

Lo svantaggio è il costo, essendo un prodotto commerciale ha costi relativamente alti.

PHP

http://www.php.net/

PHP: Hypertext Preprocessor", è un linguaggio di scripting general-purpose Open Source molto utilizzato, è specialmente indicato per lo sviluppo Web e può essere integrato nell'HTML. La sua sintassi è basata su quella di C, Java e Perl, ed è molto semplice da imparare. L'obiettivo principale del linguaggio è quello di permettere agli sviluppatori web di scrivere velocemente pagine web dinamiche, ma con PHP si possono fare molte altre cose. PHP è un linguaggio di scripting server-side. Si può vedere come un "plug-in" di un web server che aggiunge al server capacità di interpretare anche il codice PHP. Un documento PHP ha l'estensione .php e permette di eseguire operazioni come l'inserimento di dati in un database o ritornare al browser i risultati di una query ad un database sotto forma di pagina HTML.

Funziona con server web Apache su Linux e anche su IIS. Il grande vantaggio di PHP è che è completamente gratuito e scaricabile da Internet. Si può connettere con il database MySQL.

```
<HTML>
<HEAD>
<TITLE>Today's Date</TITLE>
</HEAD>
<BODY>
<P>Today's Date (according to this Web server) is
<?php
  echo( date("l, F dS Y.") );
?>
</BODY>
</HTML>
```

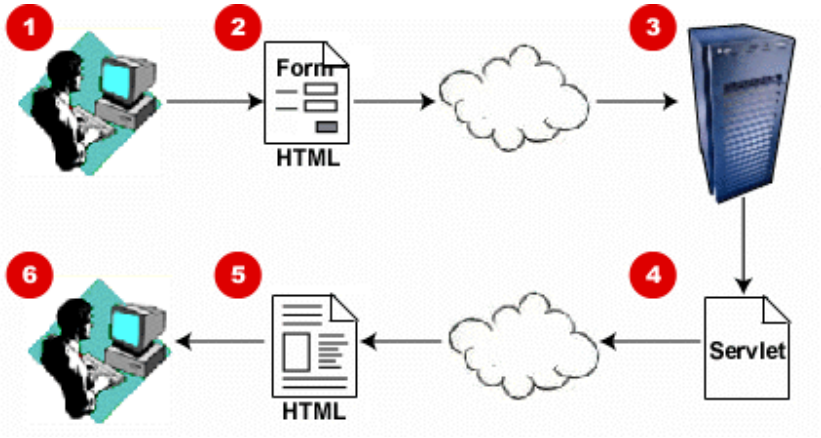
```
<HTML>
<HEAD>
<TITLE> Esempio PHP </TITLE>
</HEAD>
<BODY>
<P>

<? echo "questo è un <B>test</B>!"; ?>

</P>
</BODY>
</HTML>
```

Servlet (Java/Javascript)

Un servlet e' un programma Java incluso in un server web. I servlet sono da molti considerati come i successori dei CGI. Hanno la caratteristica, rispetto ai CGI, di essere eseguiti nello spazio di processo del web server, quindi ad ogni esecuzione di un servlet non viene creato un nuovo processo e quindi non appesantice il carico della macchina server. Inoltre, i servlet rimangono attivi su piu' sessioni e possono comunicare direttamente con altri servlets. Sono portabili su piu' piattaforme e su piu' server web.



Recentemente, ai servlet si è affiancata un'altra tecnologia, **JavaServer Pages (JSP).** E' un linguaggio di scripting (simile ad ASP) che permette di separare la parte dinamica delle pagine dall'HTML statico.
La parte dinamica viene inclusa nei delimitatori **<%** e **%>**. I file hanno estensione .jsp e risiedono sul server assieme alle normali pagine HTML. La caratteristica di JSP è che è un meccanismo per costruire servlet. Infatti, ogni pagina JPS viene convertita in un servlet standard, dove la parte HTML statica viene spedita nello standard output. Questa trasformazione avviene solitamente la prima volta che la pagina viene richiesta dall'utente il quale riceverà la pagina con un piccolo ritardo causato dal tempo di compilazione. JSP prevede tre tipi di costrutti: **elementi di scripting, direttive** e **azioni.** Gli elementi di scripting permettono di specificare il codice Java che diverrà parte del servlet risultante, le direttive permettono un controllo sulla struttura del servlet e le azioni permettono di specificare le componenti esistenti da usare oppure permettono di controllare l'engine JSP. Per eseguire script JSP occorre usare il server web Tomcat di Apache (http://www.apache.org)

Un Esempio di Servlet

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class SomeServlet extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {

// Use "request" to read incoming HTTP headers (e.g.
cookies)
// and HTML form data (e.g.data the user entered and
submitted)

// Use "response" to specify the HTTP response line
and headers
// (e.g. specifying the content type, setting
cookies).

    PrintWriter out = response.getWriter();
// Use "out" to send content to browser
  }
}
```

Tecnologie Web Client Side

E' la parte di programmazione che coinvolge il browser web ed è quindi dipendente dal tipo (e versione) del browser usato. Alcuni linguaggi client-side non sono supportati da alcuni browser – ad es. VBscript è supportato solo la IExplorer. Uno standard è Javascript - ECMA e ISO. Uno script può comportarsi in modo diverso nei vari browser web.

Tipicamente la operazioni effettuate client side sono di interazione stretta con l'utente – ad es. rollover sui tasti o il controllo di dati inseriti tramite FORM (moduli).

Generalmente abbiniamo alla programmazione client side compiti semplici e di interazione immediata con l'utente.

Abbiamo due casi: linguaggi di scripting integrati con codice HTML (Javascript, VBScript) oppure veri e propri programmi che vengono scaricati sulla macchina client ed eseguiti (Applet, Plug-in).

Linguaggi di Scripting

Le ultime generazioni di browser oltre a parserizzare codice HTML, sono stati estesi per interpretare codice Javascript (nel caso di Internet Explorer anche Visual Basic Script) che, integrato con il codice HTML di una pagina web, permette di rendere tale codice *dinamico*.

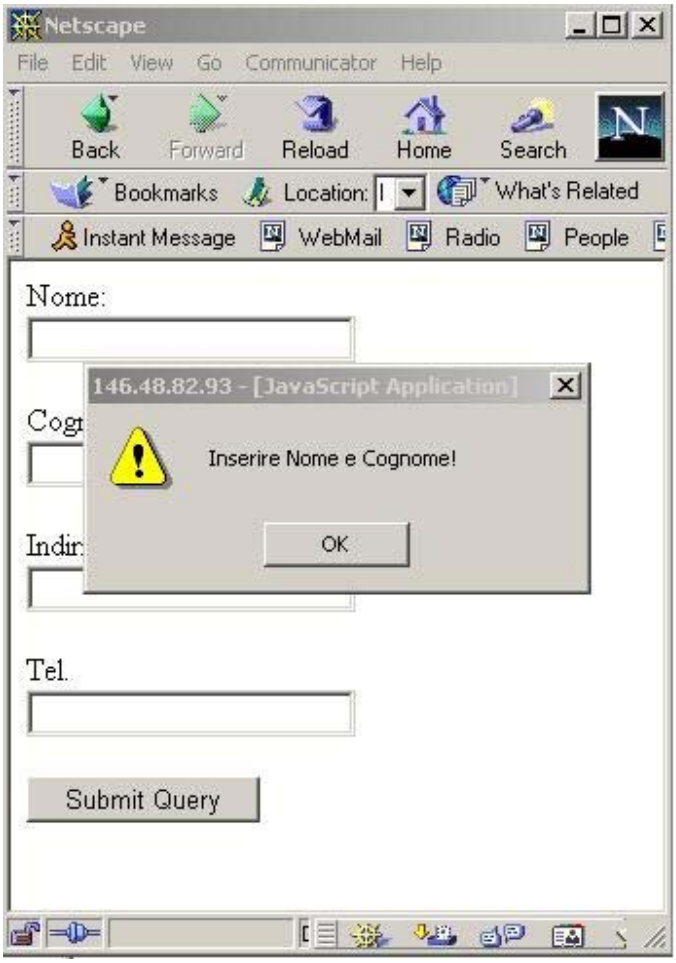
Javascript è il linguaggio di scripting sicuramente piu' usato essendo compatibile con i piu' diffusi browser web. E' un linguaggio basato su Java e come tale ha tutti i principali costrutti di un linguaggio di programmazione object-oriented. Inoltre, è un linguaggio basato su eventi, ovvero prevede dei costrutti per riconoscere eventi che l'utente può provocare sulla pagina (ad esempio può rilevare che l'utente ha premuto un pulsante di una form o che la pagina e' stata caricata dal browser).

Esempio:

Supponiamo di voler preparare una Form HTML per l'inserimento di dati da parte dell'utente. Vogliamo anche prevedere dei controlli affinché i dati inseriti siano corretti. Possiamo quindi scrivere una funzione javascript che rilevi l'evento *submit* della form da parte dell'utente e controlli i valori dei campi inseriti.

Si impedisce la sottomissione della form in caso di mancato inserimento del nome e del cognome dell'utente.

Esempio pagina HTML con controlli inserimento dati in Javascript

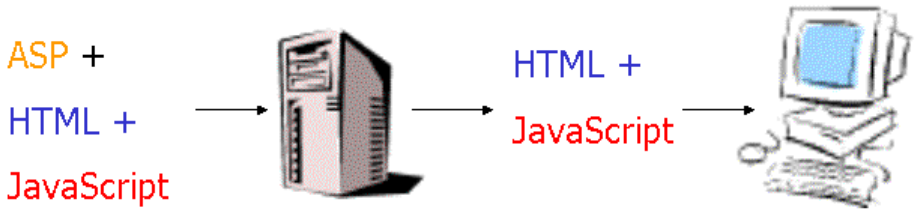


Sorgente HTML/Javascript della pagina

```
<HTML>
<HEAD>
<SCRIPT language=javascript>
 function CheckDati()
 {
  if ((document.dati.Nome.value == "") ||
(document.dati.Cognome.value == ""))
   {
   alert("Inserire Nome e Cognome!");
   return false;
   }
 else  return true;
}
</SCRIPT>
</HEAD>
<BODY>
<FORM name="dati" action="submitdata.exe"
OnSubmit="return CheckDati();">
Nome:
<BR>
   <INPUT name="Nome" TYPE=TEXT>
<P>
Cognome:
<BR>
   <INPUT name="Cognome" TYPE=TEXT>
<P>
Indirizzo:
<BR>
   <INPUT name="Indirizzo" TYPE=TEXT>
<P>
Tel.
<BR>
   <INPUT name="Tel" TYPE=TEXT>
<P>
   <INPUT TYPE=submit>
</FORM>
</BODY>
```

Le applicazioni web complesse anche se realizzate con tecnologia server side, in generale implementano comunque una parte di computazione client-side in Javascript per effettuare i controlli, ad esempio, sull'inserimento dei dati da parte dell'utente o per rilevare eventi quali il caricamento di una pagina. In questo modo si possono attivare delle risposte da parte del server o del browser stesso. Quindi in generale avremo pagine dinamiche che intervallano

codice HTML con qualche linguaggio di script server-side (ad es ASP) e con codice di script client side Javascript. Il server web interpreta la parte di script server side restituendo al browser il codice HTML + Javascript che viene visualizzato.



Applet

Gli applet sono veri e propri programmi che vengono scaricati dal browser web in locale sulla macchine dell'utente, dove vengono eseguiti. Gli applet Java possono eseguire animazioni interattive, calcoli e altri compiti senza che l'utente debba spedire i dati al server web.

In questo modo si alleggerisce il server dal peso della computazione che viene spostato sulla macchina client. Gli Applet sono codice eseguibile (byte code) dalla Java Virtual Machine (JVM) che è implementata sui browser web. Gli applet hanno delle restrizioni per garantire la privacy e le sicurezza degli utenti (ad esempio non possono scrivere sul disco locale e non possono spedire dati a server web che non sia quello di partenza).

Gli Applet possono essere inclusi in una pagina HTML con il tag <APPLET>, in modo simile a come sono incluse le immagini

Codice Java APPLET

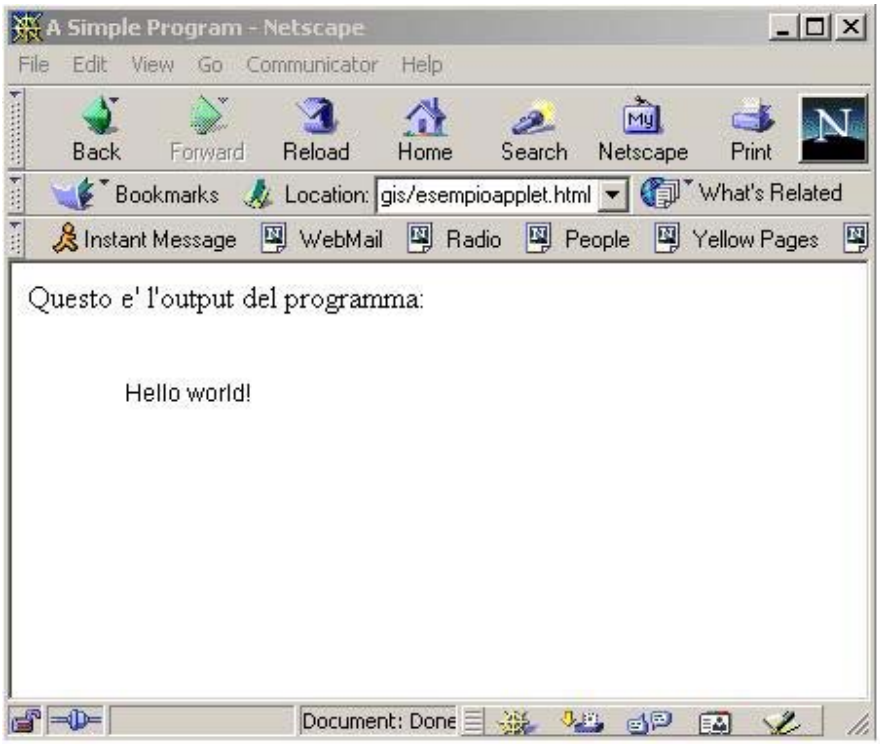
```
import java.applet.Applet;
import java.awt.Graphics;

   public class helloworld extends Applet {
              public void paint(Graphics g)
{
                  g.drawString("Hello
world!", 50, 25);
               }
    }
```

Sorgente HTML della pagina contenente l'APPLET

```
<HTML>
   <HEAD>
   <TITLE> Hello World </TITLE>
   </HEAD>
  <BODY>
   Questo e' l'output del programma:
   <P>
    <APPLET CODE="helloworld.class" WIDTH=150
HEIGHT=25>
    </APPLET>
 </BODY>
 </HTML>
```

Pagina risultante dall'esecuzione dell'APPLET



Plug-in

Le applicazioni *plug-in* sono programmi che possono facilmente essere installati e usati come parte del web browser. Un plug-in e' riconosciuto automaticamente dal browser e le sue funzioni sono integrate nel file HTML che viene presentato. Per Internet Explorer i plug-in sono sostituiti da oggetti ActiveX.

Tra i plug-in piu' diffusi:

- Adobe Acrobat per visualizzare documenti .pdf
- RealNetwork Media Player
- Macromedia Shockwave
- Macromedia Flash

I plug-in, rispetto agli applet che vengono scaricati automaticamente dal server sul client, richiedono un intervento esplicito dell'utente che deve preoccuparsi di installarlo sul suo PC. Una volta installato, il plug-in risiede in modo permanente sul client (l'utente puo' comunque disinstallarlo). I plug-in

riconoscono le estensioni dei file a loro associati e si attivano quando questi file vengono caricati dal browser.
Esempio di sorgente HTML con incorporato un plug-in:

```
<HTML>
<HEAD>
    <TITLE> Esempio Plug-in </TITLE>
</HEAD>
<BODY>

<object id="objACGM"
   classid="clsid:F5D98C43-DB16-11CF-8ECA-
0000C0FD59C7"

codebase="acgm/acgm.cab#version=6,0,10,0"
            WIDTH="500" HEIGHT="450">
<param name="FileName"
value =
http://Maps.Intergraph.Com/GwenDemo/CGMs/None.
cgm>

<embed name="objACGM"
     src =
http://Maps.Intergraph.Com/GwenDemo/CGMs/None.
cgm
      width="500" height="450">
</object>
</BODY>
<HTML>
```

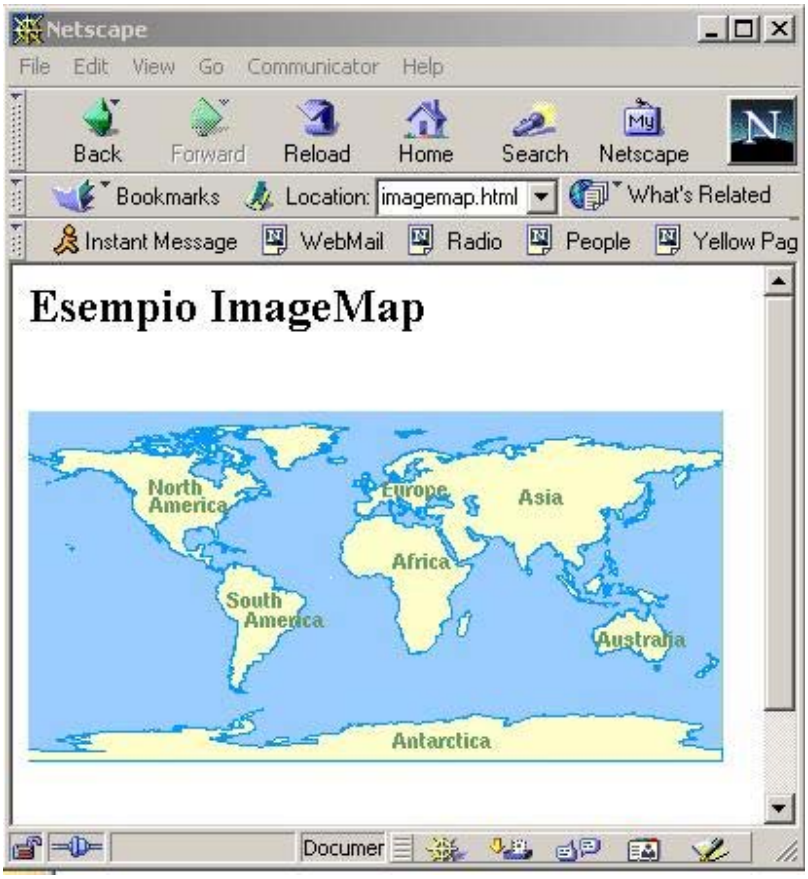
ImageMap

Una *imagemap* non è una vera e propria tecnologia client-side, in quanto e' HTML standard. Pero' la citiamo qui perché molto usata in ambito WebGIS e consiste in una immagine grafica definita in modo tale che un utente puo' cliccare in aree linkate a URL diverse.

Si definiscono delle *aree sensibili* in termini delle loro coordinate e si specifica quale e' la URL associata al click dell'utente in quella data area. Le imagemap si possono creare con appositi tool come MapEdit. Una lista si trova ad esempio all'indirizzo:

http://download.html.it/categorie/start/53/windows/image-map/

Esempio di pagina con imagemap



Sorgente HTML

```
<HTML>
<HEAD>
<TITLE> Esempio ImageMap </TITLE>
</HEAD>

<body LINK="#003366" VLINK="#003366">

<H2> Esempio ImageMap </H2>

<img name="mondo" src="mappina.gif" border="0"
usemap="#mappina">

<map name="mappina">
<area shape="circle" alt="Nord America"
coords="79,66,40" href="nordamerica.html" title="Nord
America">
<area shape="rect" alt="Sud America"
coords="93,105,156,171" href="sudamerica.html"
title="Sud America">
<area shape="rect" coords="160,79,231,148"
href="africa.html" title="">
<area shape="rect" coords="163,39,225,78"
href="europa.html" title="">
<area shape="rect" coords="234,34,355,112"
href="asia.html" title="">
<area shape="rect" coords="262,114,355,171"
href="australia.html" title="">
<area shape="rect" coords="8,179,352,199"
href="antartide.html" title="">
<area shape="default" nohref>
</map>
</BODY>
</HTML>
```

AJAX[3]

AJAX, (Asynchronous JavaScript And XML) non è una nuova tecnologia **ma** di un concetto utilizzato per sviluppare applicativi avanzati e particolari quali Gmail, Google Maps o Google Suggest. Si tratta di un utilizzo asincrono di Javascript e XML, per permettere ad un client di richiamare informazioni lato server in modo veloce e trasparente.

Vuole sostituire tecnologie come Adobe-Macromedia Flash o Java (con le applet) ed è stato sfruttato da Google, ad esempio con il famoso google mail.

AJAX è di fatto un oggetto specifico: **XMLHttpRequest**. A seconda del browser usato prende nomi differenti o viene richiamato in maniera differente.
Nel caso di Internet Explorer, ad esempio, questo oggetto è restituito da un ActiveXObject mentre nei browsers alternativi più diffusi (Mozilla, Safari, FireFox, Netscape, Opera ed altri) `XMLHttpRequest` è supportato nativamente, cosa che dovrebbe accadere anche per IE dalla versione 7.

Questo oggetto permette di effettuare la richiesta di una risorsa (con HTTP) ad un server web in modo indipendente dal browser. Nella richiesta è possibile inviare informazioni, ove opportuno, sotto forma di variabili di tipo GET o di tipo POST in maniera simile all'invio dati di un form.

La richiesta è asincrona, il che significa che non bisogna necessariamente attendere che sia stata ultimata per effettuare altre operazioni, stravolgendo sotto diversi punti di vista il flusso dati tipico di una pagina web.
Generalmente infatti il flusso è racchiuso in due passaggi alla volta, richiesta dell'utente (link, form o refresh) e risposta da parte del server per poi passare, eventualmente, alla nuova richiesta da parte dell' utente.
La tecnologia Ajax permette di ridurre l'attesa che trascorre tra una richiesta dell' utente e la risposta del server. Infatti, mentre l'utente è all'interno della stessa pagina le richieste sul server possono essere numerose e completamente indipendenti. Nulla infatti vieta, teoricamente, di effettuare decine di richieste simultanee al server per operazioni differenti con o senza controllo da parte del navigatore.

Vantaggi della tecnologia Ajax

Impressione di avere a che fare con pagine apparentemente più interattive, dinamiche e professionali

Librerie dedicate, facilmente integrabili in applicativi AJAX, per grafica più sofisticata.

[3] testo tratto in parte da www.html.it

Compatibilità praticamente totale con i browser più diffusi ed il supporto nativo, o integrabile, dell'oggetto `XMLHttpRequest`.
Ottenere modifiche dinamiche e leggere sulle pagine, come i suggerimenti sui campi di ricerca, in grado di suggerire parole presenti e velocizzare l'inserimento, mappe, votazioni, statistiche, amministrazione, lezioni, chat, giochi…

Svantaggi
Penalizza coloro che non dispongono di connessioni veloci in quanto devono caricare al primo accesso una mole di dati consistente rappresentata dai vari file JavaScript.
Browser devono essere aggiornati e Javascript abilitati.
L'uso dei tasti 'avanti' e 'indietro' presenti nei browser non rende su una pagina Ajax.
Aprendo direttamente una pagina ricca di interazioni asincrone non sarà possibile, nemmeno volendo, cliccare sul tasto 'indiero' del browser per tornare allo stato precedente, si verrà reindirizzati invece alla pagina precedente,
Oltre ad essere un vincolo di navigazione, questo problema è anche un vincolo per l'indicizzazione o la possibilità di segnalare ad altri la pagina visualizzata.

ESERCITAZIONE SU SERVER WEB

Esercizio 1.
Cambiare la porta di funzionamento della vostra installazione di apache, ad esempio settandola a 8080. Verificare il funzionamento dal browser. Riportare la configurazione alla porta 80.

Esercizio 2.
Creare un file di testo e salvarlo nella cartella di default del server web. Visualizzarlo dal browser web tramite il server (localhost).

Esercizio 3.
Creare una cartella WEBGIS nella cartella di default del server web. Creare un nuovo file di testo e salvarlo nella nuova cartella. Verificarne la visualizzazione dal browser web (da localhost).

Esercizio 4.
Creare la cartella CambioRoot sul disco C e salvarvi dentro un nuovo file di testo. Cambiare la document Root e settarla a CambioRoot, verificarne il funzionamento. Riportare la DocumentRoot a quella originale.

Esercizio 5.
Creare una cartella sul disco C: EsempioAlias e creare un alias EsAliasWebGis, verificarne il funzionamento.

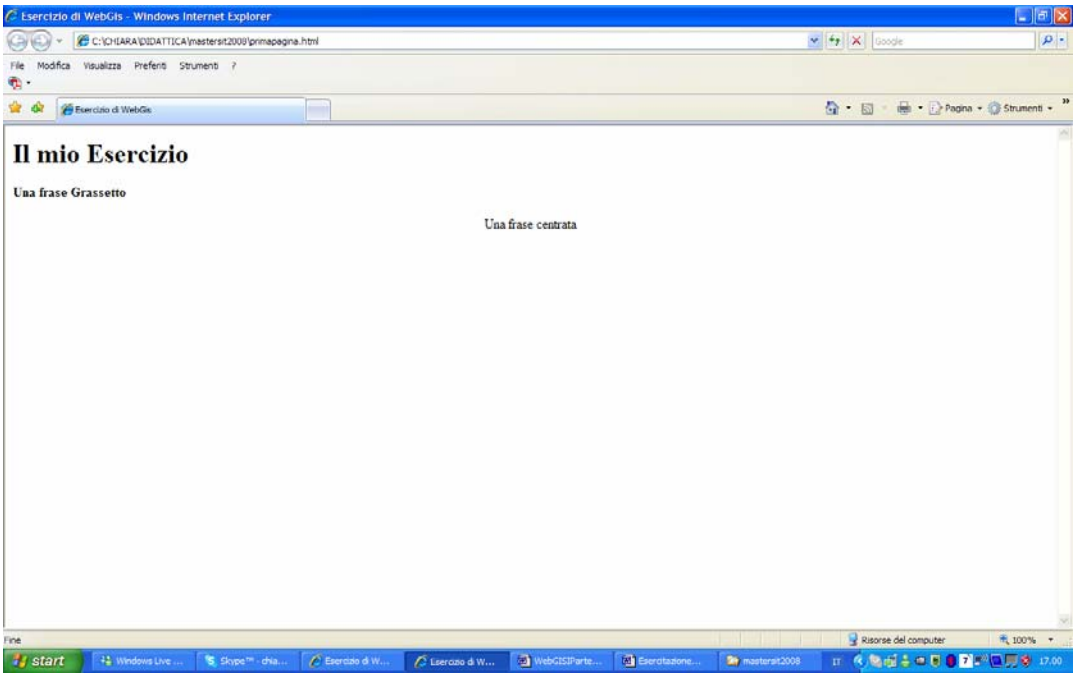
Esercizio 6.
Visualizzate il sito del vostro vicino dal vostro browser digitando il nome o l'IP address del PC del vostro vicino.

Esercizio 7.
Visualizzate i file di log di apache

Esercitazione su HTML

Esercizio 1:

1. Creare (se non esiste già) nella cartella di default di Apache (htdocs) una cartella di nome EsWebGis
2. Creare nella cartella EsWebGis una pagina HTML il cui titolo sia "Il Mio Primo Esercizio" e che visualizzi in centrato con carattere grande "Il mio Esercizio", seguito a capo dalla frase in grassetto "una frase in grassetto" e seguito ancora a capo dalla frase centrata "una frase centrata" (usando blocco note – *attenzione a selezionare "tutti i file" quando si salva*). Esempio di pagina risultante:



3. Dal browser web, aprire la url (digitando la URL da localhost non da "apri file"!) corrispondente alla pagina appena creata e verificarne la corretta visualizzazione.
4. cambiare il colore di sfondo della pagina e dei font della frase centrata.

Esercizio 2:
Creare nella cartella EsWebGis un'altra sottocartella chiamata SubEsWebGis e create all'interno un nuovo file dal titolo "Il Mio Secondo Esercizio" contenente una elenco di tre elementi. Visualizzare la pagina dal browser web (sempre tramite il server web).

Esercizio 3:
Aggiungere al secondo esercizio una immagine a vostra scelta (anche scaricata da web). Visualizzare l'esercizio.

Esercizio 4:

Realizzare in HTML la seguente tabella:

1-1	1-2	1-3	1-4
2-1	2-2	2-3	2-4
3-1	3-1	3-3	3-4

Esercizio 5:

Realizzare la home page del vostro sito personale di fornitura su Internet di dati geografici. La home page dovrà contenere il nome del sito (ad es: "GeoMaster"), link a sezioni del sito (chi siamo, download dati, visualizza mappe, contattaci etc...), una breve frase di presentazione (di vostra fantasia). Il layout del sito dovrà utilizzare una tabella invisibile con una colonna per i link e una per i contenuti. Font e colori a vostra scelta.

Esercizio 6.

Creare una nuova sezione del sito "Esprimi le tue preferenze" dove raccogliere il feedback degli utenti che apra una nuova pagina: preferenze.html. Creare quindi la nuova pagina preferenze.html linkata (potete copia-incollare una delle pagina precedenti) che contenga una FORM (come action non inserire nulla).
La form dovrà realizzare un modulo per l'interazione utente dove vengono richiesti nome e cognome (campo INPUT di tipo TEXT), città di nascita (campo di selezione tra alcune città a vostra scelta), formato dei dati preferito (4 o piu' checkbox tra formati di dati geografici possibili).

Esercizio 7.

Cercate sul web un'immagine dell'Italia divisa in regioni non troppo piccola. Inserirla in una pagina HTML e, con l'aiuto di uno strumento come ad esempio *Mapedit*, realizzare una imagemap in modo tale che cliccando su una regione si raggiunga una pagina relativa alla regione stessa. Ad esempio cliccando sulla toscana si raggiungerà la pagina toscana.html che conterrà una serie di informazioni che potete trovare su web, come ad esempio una descrizione, l'elenco delle province, la popolazione etc. Realizzare almeno due o tre regioni.