Scripting
Applicazioni server side

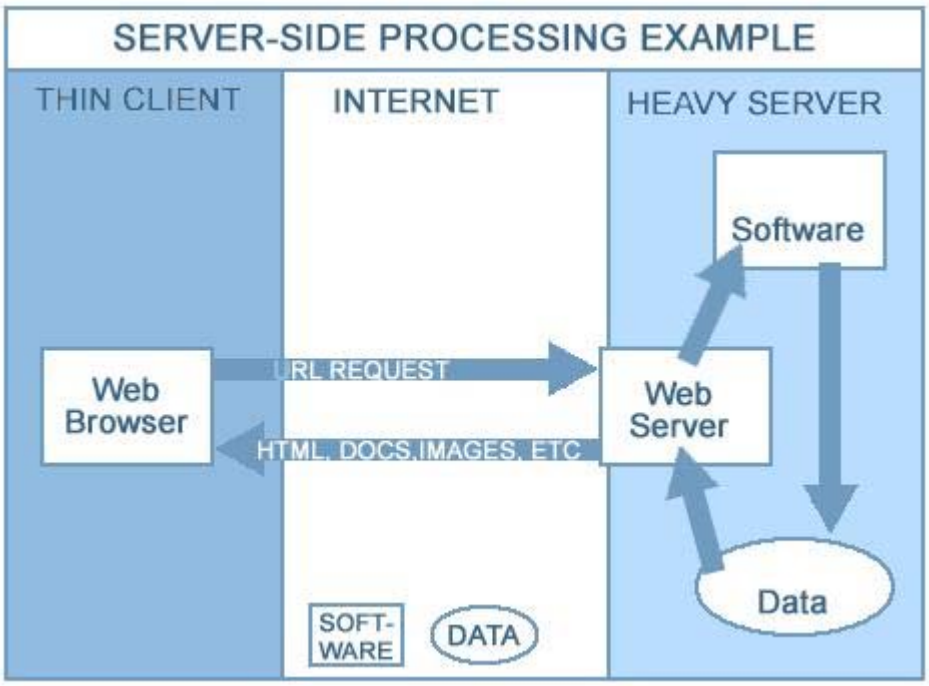
Con tecnologia *Web server-side* si indica, un insieme di meccanismi che permettono al *server Web* di elaborare informazione. Il *server Web* non si limita solo a rispondere a richieste HTTP restituendo documenti **HTML**, ma é in grado di eseguire anche una fase di elaborazione dei dati. L'utente puo' quindi interagire con il *server* ad esempio sottomettendo dati che il *server* elabora e restituisce poi la risposta sotto forma di pagina **HTML**. Un tipico caso é 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. 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**, **JSP** e **PHP**, **ISAPI**, **NSAPI**, **Servlet**. Queste si distinguono tra loro per sfruttare in modo diverso l'interazione col *Web server*: alcune di esse sfruttano ***Script*** all'interno delle pagine **HTML** (tecnica detta di Inclusione Lato ***Server***) mentre altre utilizzano dei veri e propri programmi, che passano informazioni al *Web server* sfruttando le *API* (*NSAPI*, *ISAPI*) dei *Web server*.

Server Side Include	***API***
ASP (*Microsoft*)	**ISAPI (Microsoft)**
PHP	**NSAPI (Netscape)**
JSP (java server pages)	**Servlet**

Esecuzione di processi lato server

Il passaggio di semplici documenti **HTML** tra il *server* e il *client* non permette lo sviluppo di applicazioni *Web* complesse che coinvolgano una fase di elaborazione oltre che di passaggio di dati. Applicazioni complesse necessitano di un elevato grado di *processing* (*query*, analisi, transazioni...). Dove deve essere eseguita tale computazione? Solo dalla parte server? O è meglio spostare parte del peso della computazione anche sul *client*? E se si, quanto?



Il modello ***Client-Server*** permette la condivisione di informazioni e il livello di elaborazione è

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*. Globalmente quindi una applicazione *Web* può essere realizzata con tecnologia *server side* quando il peso della computazione risiede tutta sul *server*, oppure con tecnologia *client-side* quando la computazione avviene principalmente sul *browser*. Generalmente le applicazioni *Web* complesse usano entrambe queste strategie.

Tecnologie Web Server-Side e Client-Side

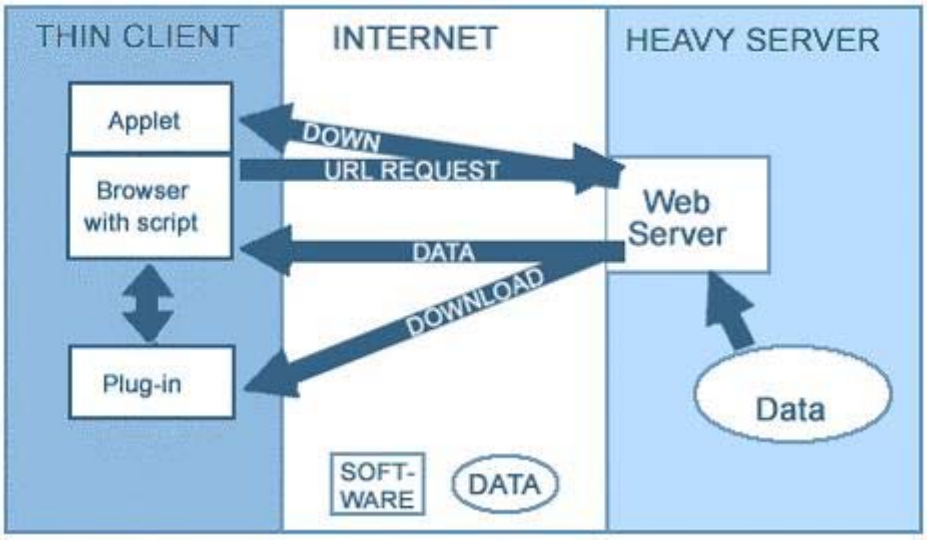
I vantaggi di una tecnologia *server side* sono quelli di avere un sito centralizzato e quindi:

1. più facile da mantenere e tenere aggiornato;
2. più semplice da controllare e gestire gli accessi;
3. avere un *server* potente permette:
    - funzionalità 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, che può causare:

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

Usare una tecnologia *client side* significa spostare parte o tutto il peso della computazione sul *client*. Il *client* di una applicazione *Web* è il *browser*.

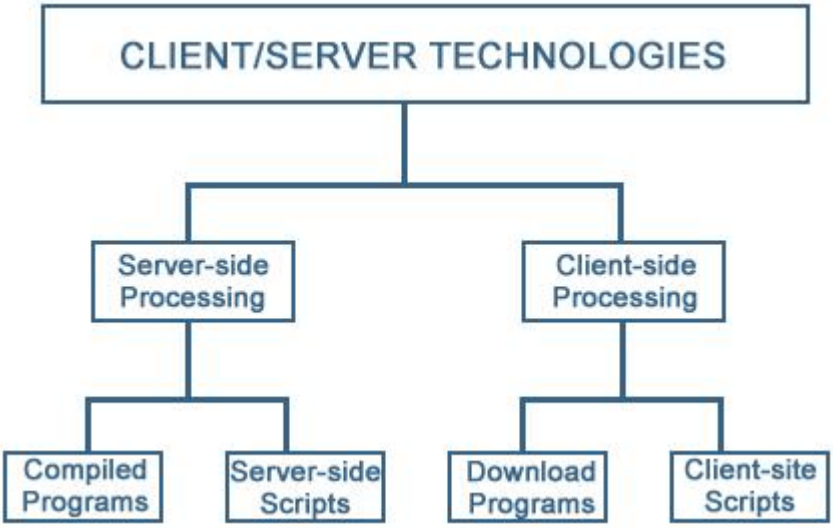


I *browser Web* sono chiamati thin client cioè *client* che non hanno potere di calcolo, ma solo capacità di visualizzazione. Comunque, i *browser* forniscono meccanismi per includere altre tecnologie come *Java Applets*, *Active X* e *Plug-in*. I vantaggi di usare una tecnologia *client side* stanno 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 eccetera);
- *dowloading* iniziale di datasets anche grandi
- il *computer client* può assere weak cioè non avere grossa potenza di calcolo



Negli approfondimenti si discutono alcune tecnologie *server-side* molto note: **CGI**, **ASP**, **Servlet** e **PHP**.