Installazione di server
Struttura di un server Web - Cosa è un Web server

Un ***server*** *Web* è sostanzialmente un *software* che fornisce all'utente pagine **HTML** da visualizzare nel proprio *browser*. Il **protocollo** su cui si basa è detto **HTTP**. Tale **protocollo** stabilisce quali sono le regole per richiedere documenti ad un ***server***. Questo **protocollo** è diviso in due parti: la richiesta del ***client*** e la risposta del ***server***. Il **protocollo** è basato completamente su testo ASCII, ma non stabilisce il formato della risorsa restituita al ***client***. In altre parole è compito del *browser* ricevere il documento (pagina HTML, immagini gif, eccetera) e visualizzarlo correttamente.

Il server HTTP

Un ***server*** HTTP è il programma (anche detto demone) che gestisce e soddisfa le richieste **HTTP** ricevute su di un particolare canale di un elaboratore. In pratica è il programma che realizza le potenzialità di **HTTP** e che ne implementa le caratteristiche.

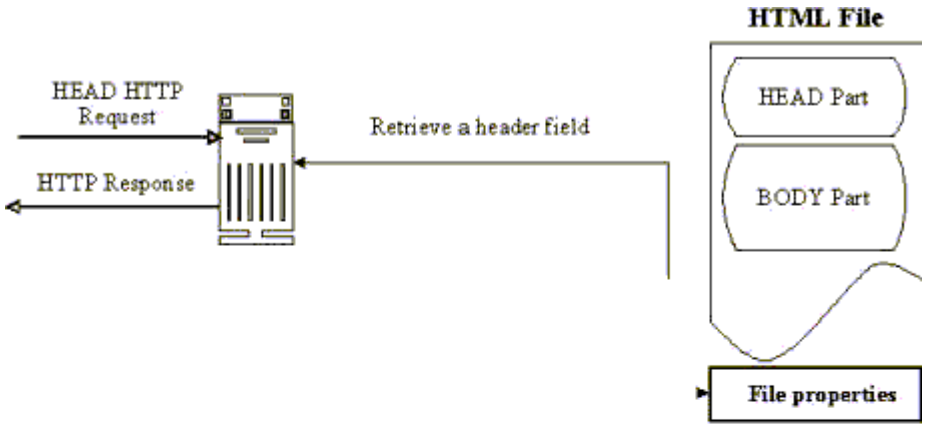
Questo tipo di programmi rimangono in una situazione di attesa fino a quando non vengono risvegliati dalla presenza di un segnale sulla porta loro assegnata, che in genere è la 80. A questo punto il ***server*** legge il segnale e ne controlla la sintassi (definita dal protocollo **HTTP**). Se la sintassi viene riconosciuta come valida, il demone, interpretata la richiesta, cerca di soddisfarla; sia che il ***server*** riesca ad accedere ai dati richiesti, sia che non ci riesca, comunque esso manda una risposta o contenente i dati richiesti o indicante la situazione di errore.

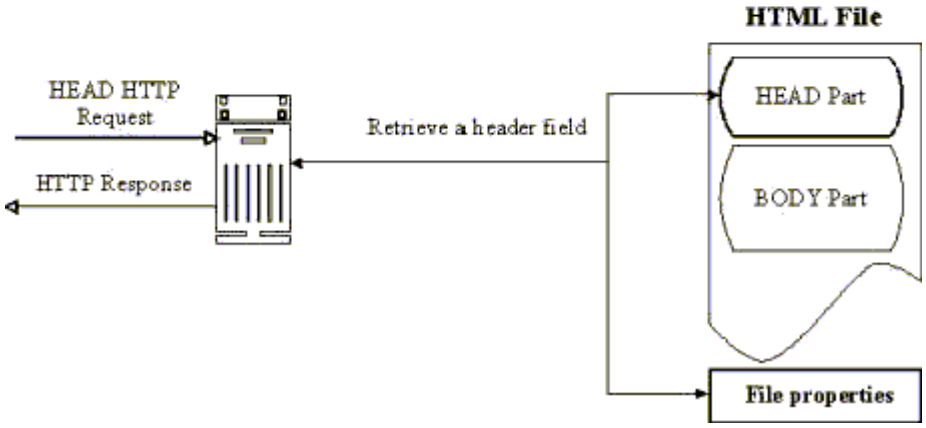
Questo funzionamento non è comunque caratteristico dei soli HTTPd (demoni **HTTP**) ma anche di molti ***server*** che implementano altri protocolli come **FTP** (*File Transfer Protocol*), *Gopher* o NNTP (*Network News Transfer Protocol*). Al ***server*** viene comunque assegnato uno spazio fisico sull'*hard disk* in cui può rintracciare i documenti richiesti, e che viene identificato come spazio del *Web* o Web space. Un documento, quindi, per poter essere distribuito, deve essere contenuto in questo spazio altrimenti il ***server*** non riesce a recuperarlo.

Come succede con la gran parte dei sistemi *client-server*, quando il ***server*** riceve una richiesta si attivano le seguenti attività:

- genera un sub-**processo** per rispondere alla richiesta;
- allo stesso tempo gli altri processi continuano ad ascoltare sulla porta 80 per nuove richiesta;
- il sub-**processo** generato risponde alla richiesta.

Compito del *Web* ***server*** è anche quello di fare da *gateway* tra i dati ricevuti con una richiesta e una **applicazione**. In questa maniera è possibile accedere ad archivi, piuttosto che modificare *run time* un *file* in dipendenza ai dati forniti oppure per poter aggiornare i dati di una pagina **HTML** (pensiamo ad una pagina che mostra il numero di accessi al *Web* ***server***). Alcune implementazioni di HTTPd permettono di cifrare i messaggi, per poter effettuare delle transizioni di dati in maniera sicura (protocollo **SSL**) oppure richiedere al ***client*** un'autenticazione, per poter selezionare gli utenti a cui concedere l'accesso.





Questo **protocollo** è diviso in due parti: la richiesta del ***client*** e la risposta del ***server***. Il **protocollo** è basato completamente su testo ASCII.

Richiesta di un *client*

La richiesta è una linea di testo divisa in 3 parti:

1. tipo di richiesta.
2. **URL**.
3. Protocollo usato.

Le richieste possibili al punto 1 sono: ***GET***, ***POST***, ***HEAD***, *PUT*, *DEL*, *TRACE*. L'**URL** è il percorso al quale si vuole accedere nel chiedere un oggetto, il quale deve essere comprensivo del nome dell'oggetto. Ad esempio:

```
www.aipa.it/prove/pagina.asp?par=val&par2=val2
```

1. **www.aipa.it** è il **nome di dominio**;
2. **/prove/** è la cartella locale del *Web-**server***;
3. **pagina.asp** è il documento richiesto;
4. **?par=val&par2=val2** rappresenta un elenco di parametri che possono essere inviati al *Web **server***.

Formato dei messaggi HTTP

Formalmente una messaggio **HTTP** 1.0 è composto da una richiesta e una risposta:

```
Request = Request-line *(General-Header | Request-Header | Entity-Header) CR LF
[Entity-Body] Response = Status-Line *(General-Header | Request-Header | Entity-
Header) CR LF [Entity-Body] Request-Line = Method (Spc) Requested-URI (Spc) HTTP-
Version CRLF General-Header = Date | MIME-Version | Pragma Request-Header =
Authorization | From | If-Modified-Since | Referer | User-Agent Entity-Header =
Allow | Content-Encoding | Content-Lenght | Content-Type| Expires |Last-Modified
| extension-header Entity-Body = *OCTET (qualsiasi sequenza di dati codificati a
```


```
8 bit)
```

I Metodi

GET*:** serve per recuperare qualsiasi tipo di *file* (intesa come entità) identificato dall'URI specificato. Un esempio di richiesta **HTTP** di tipo ***GET, e relativa risposta può essere:

```
GET http://www.aipa.it/index.html HTTP/1.0 HTTP/1.0 200 OK
Date: Fri, 24 Jun 2002 11:33:18 GMT
Server: Apache/1.0.0
Content-type: text/html
Content-length: 3395
<html>
<head>
<title> Sito Web AIPA </title>
</head>
[...]
```

HEAD*:** è un metodo molto simile a quello del ***GET, eccetto che serve a recuperare solo gli attributi di un *file* e non il suo contenuto. È il metodo lanciato dai *browser* per controllare se il *file* richiesto non sia già presente in locale (controllando l'*header*). Se così fosse non sarebbe necessario richiederlo inutilmente al ***server***. In pratica fornisce come risposta gli stessi dati di una richiesta di tipo ***GET*** a parte l'*Entity-Body*. Per effettuare una richiesta di tipo ***HEAD***:

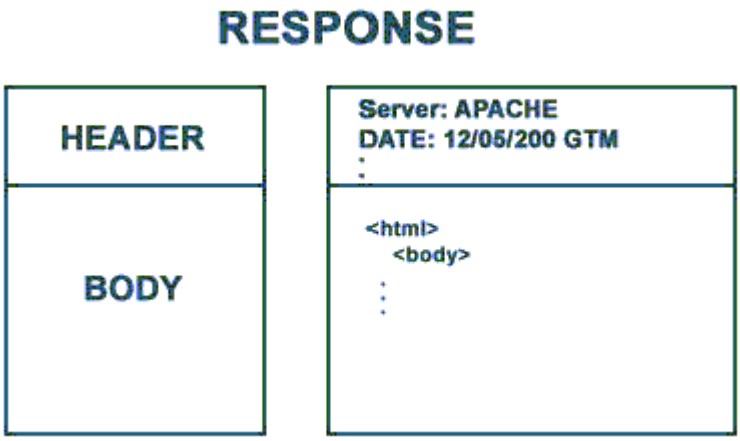
```
HEAD http://www.aipa.it/index.html HTTP/1.0 HTTP/1.0 302 Found
MIME-Version: 1.0
Server: IIS/4.0
Date: Friday, 23-Jun-02 19:40:14 GMT
Location: http://www.aipa.it/index.html
Content-Type: text/html
Content-Length: 2133
```

POST*:** questo metodo è usato per richiedere che il ***server di destinazione accetti l'entità acclusa al pacchetto di richiesta come sotto-coordinata della risorsa identificata dall'URI nella *Request-Line*. In pratica viene usato per poter passare dei parametri ad un programma di elaborazione, attivato via richiesta **HTTP**, che spesso manda una risposta dipendente dalla *query*. Questo metodo non obbliga al *Web* ***server*** a fornire una pagina oggetto come risposta, ma solo la testata, per cui è utili nel caso in

cui una richiesta non deve modificare la pagina da cui è stata lanciata.

Risposta del *server*

La prima linea parla del **protocollo** usato e restituisce un valore del ***server*** (un valore superiore a 400 indica un errore). È seguito dalla data, la versione del ***server*** e la data dell'ultima modifica dell'**URL** (permette al ***client*** di sapere se i *file* nella sua *cache* sono ancora validi). *Content-Length* è la lunghezza della risposta (richieste a *script* ***CGI*** non forniscono queste informazioni) e *Content-Type* comunica al *Web* ***client*** il tipo di ***MIME*** usato nella risposta (testo, HTML, immagini ...).



Un errore: proviamo a connetterci ad un ***server*** HTTP tramite *telnet*, ed osserviamone la risposta (in italico quello che risponde il sistema).

```
>telnet www.aipa.it 80 Trying 195.213.25.18... Connected to www.aipa.it get /
HTTP/1.0 <invio> HTTP/1.1 501 Method Not Implemented Date: Mon, 27 Sep 1999
21:22:03 GMT Server: Apache/1.3.3 Connection: close
```

Come potete notare, l'*header* rende inutile alcuna spiegazione, ed alla fine della connessione verremo buttati fuori col codice di errore 501. L'**HTTP** è un **protocollo** molto semplice, come è ben evidenziato da questi esempi:

```
>telnet www.iol.it 80 Trying 195.213.125.118... Connected to www.iol.it GET / <
invio > [segue sullo schermo il contenuto della pagina di default del sito iol ]
```

Cosa è successo dentro il ***server*** *Web*? Vi siete connessi con il *telnet* alla porta 80 di www.iol.it (**indirizzo** *IP* 195.213.125.118) (la 80 è la porta standard del ***server*** HTTP). Il ***server*** è stato scritto per rispondere a delle richieste e voi avete scritto ***GET*** / seguito da *return*.

Codici di errore: riportiamo in seguito i codici di errore che restituisce un ***server*** HTTP nelle varie circostanze.

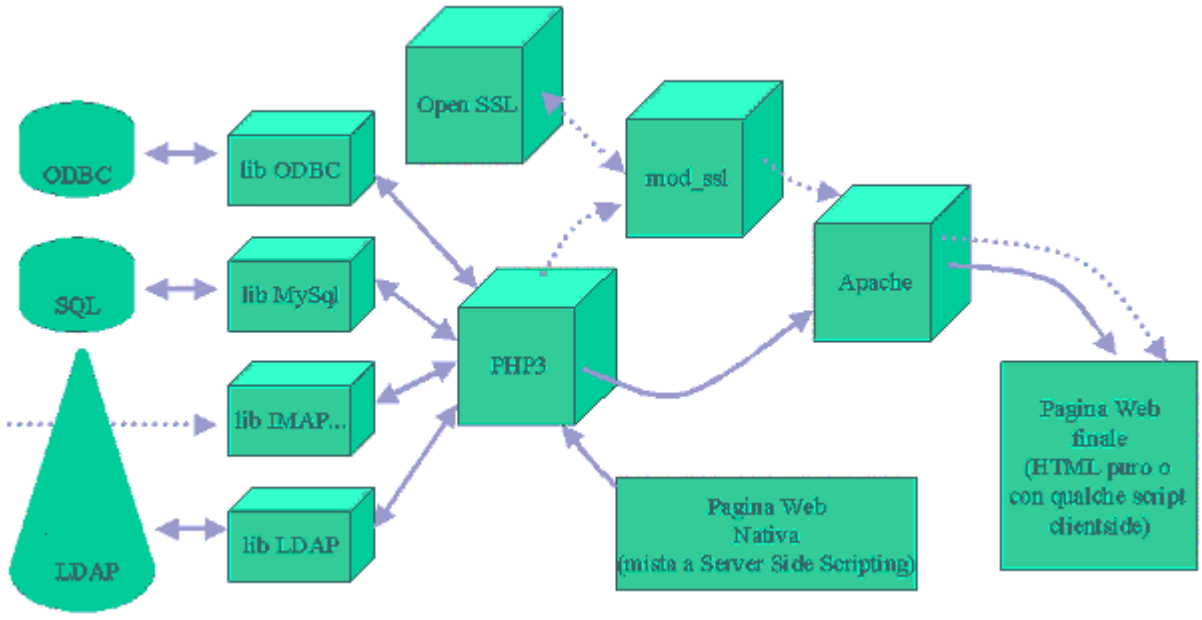
Codice	Descrizione
400	*Syntax Error*
401	Non autorizzato. Richiesta autenticazione
402	*Browser does not support required encryption method. Challenge / Response is being used*
403	Rifiutato. Esempio, cercare di aprire una pagina **SSL** con un *browser* non abilitato
404	*File* non trovato. *Link* errato o spazi nel nome di *directory*

405 *Method Not Allowed*. Problemi con Mime
406 *Not Acceptable*
407 *Proxy Authentication Required*
412 *Precondition Failed*
414 *Request-URI Too Long*. O si è sotto attacco *hacker*, o vi sono problemi col comando ***POST***
500 *Internal Error*. Esempio, lo *user* ***Anonymous*** non ha diritti **ACL** sui *file* che deve accedere
501 *Not Implemented*
502 *Bad Gateway*. Esempio, cercare di utilizzare un IDC con **DNS** sbagliato

Una definizione di ***server*** *Web* siffatta però rischia di essere allo stato attuale riduttiva: i moderni *Web* ***server*** oltre che restituire informazioni sono in grado di comunicare con *Application Server*, *software* in grado di elaborare informazioni, restituite poi al ***client*** sotto forma di pagine **HTML** dette dinamiche. La possibilità elaborativa è divenuta essenziale poiché permette al *Web* ***server*** di funzionare come interfaccia *Web* (e quindi indipendente dalla piattaforma) verso un qualunque sistema informativo. In altri termini un ***client*** chiede elaborazioni remote ad un sistema complesso, che opera su una qualunque piattaforma, veicolando le richieste attraverso una interfaccia semplice, diffusa e generica: il ***server*** *Web*.

Server Web: Apache

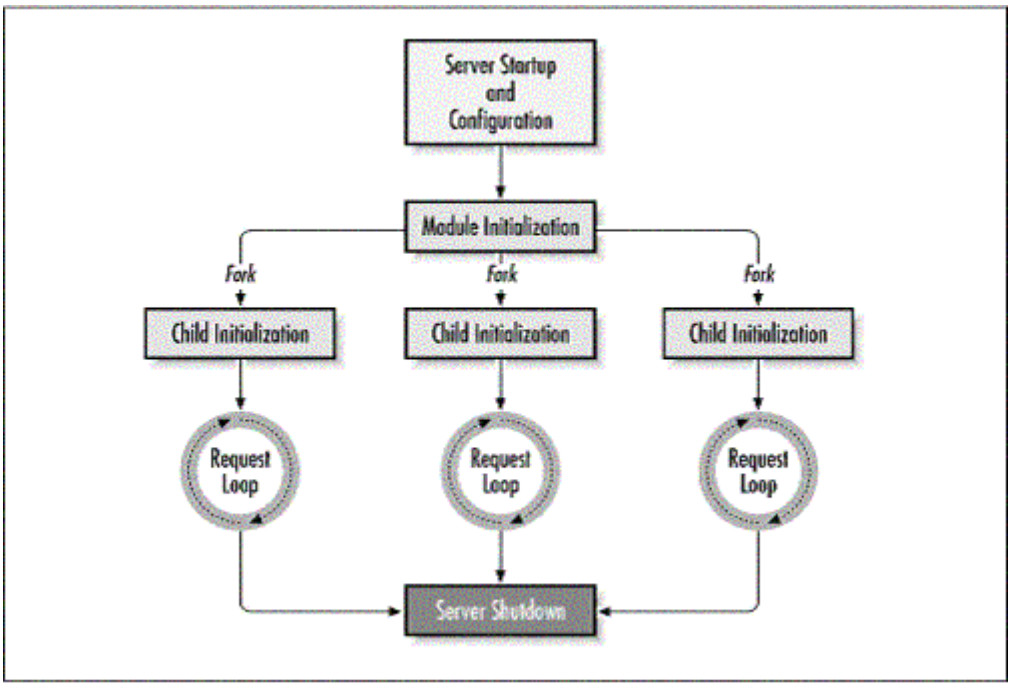
Apache (il cui *team* di sviluppo è formato da volontari, noti come *Apache Group*, www.apache.org) è nato come sostituto per il *Web* ***server*** HTTPd 1.3 sviluppato dal NCSA (*National Center for Supercomputing Applications*), inglobandone le caratteristiche, risolvendone i problemi ed implementando nuove caratteristiche. È il prodotto al momento più diffuso sulla rete, contando il doppio circa delle installazioni del suo diretto concorrente, **IIS** di *Microsoft*.



Le ragioni di tale successo vanno attribuite alle sue caratteristiche di flessibilità ed affidabilità, ma ancora di più alla filosofia commerciale che lo contraddistingue: è un prodotto open e liberamente distribuibile, così come il sistema operativo che lo ospita, *Linux*: in definitiva oltre alla flessibilità permette di contenere i costi di gestione, nonostante possiede caratteristiche in tutto e per tutto simili (se non superiori) alla diretta concorrenza.

Apache è un *Web* ***server*** che gestisce il protocollo **HTTP**, gira come un **processo** *stand-alone*, senza

cioé chiedere l'appoggio ad altre applicazioni né direttamente all'utente. Per poter fare ciò, ***Apache***, una volta che è stato avviato, crea dei sottoprocessi (comunemente detti processi *children*) per poter gestire le richieste: questi processi, non interferiscono mai con il **processo** padre, ma può succedere l'opposto: quando il **processo** padre viene terminato, ad esempio con un segnale di *stop*, anche i *children* saranno terminati.

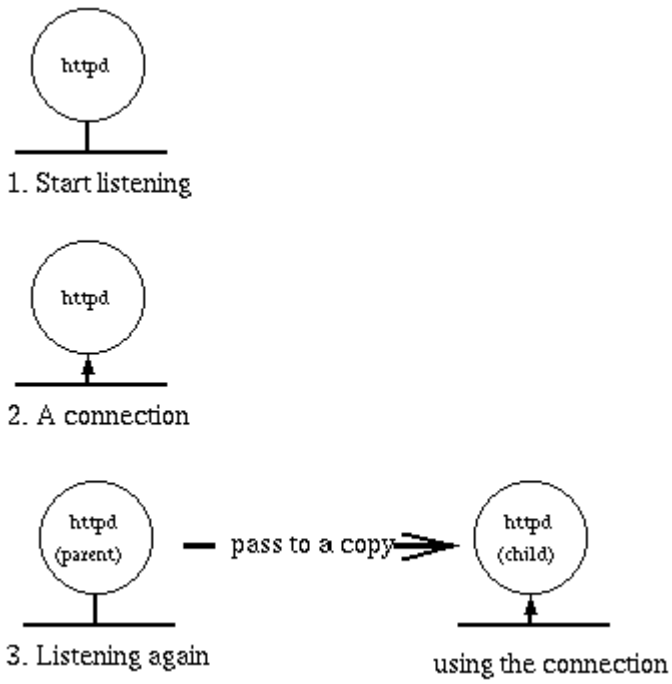


Un tipico albero dei processi di ***Apache*** è qualcosa di simile a quello illustrato nella tabella seguente: lo *user* è il proprietario del **processo**, ed il primo *threed* (*root*) è il **processo** lanciato per avviare il *Web* ***server***.

USER	*PID*	*%CPU*	*%MEM*	*SIZE*	*RSS*	*TTY*	*STAT*	*START*	*TIME*	*COMMAND*
root	203	0.01	1.01	4952	720	?	S	17.20	0.03	`/usr/sbin/apache`
user	212	0.00	2.03	5012	1456	?	S	17.20	0.00	`/usr/sbin/apache`
User	213	0.00	2.02	5008	1424	?	S	17.20	0.00	`/usr/sbin/apache`
user	214	0.00	0.00	4976	0	?	SW	17.20	0.00	`\_(apache)`
user	216	0.00	0.00	4976	0	?	SW	17.20	0.00	`\_(apache)`
user	473	0.00	1.06	4976	1072	?	S	18.05	0.00	`/usr/sbin/apache`
user	477	0.00	1.06	4976	1076	?	S	18.05	0.00	`/usr/sbin/apache`
user	478	0.00	2.04	5012	1544	?	S	18.05	0.00	`/usr/sbin/apache`

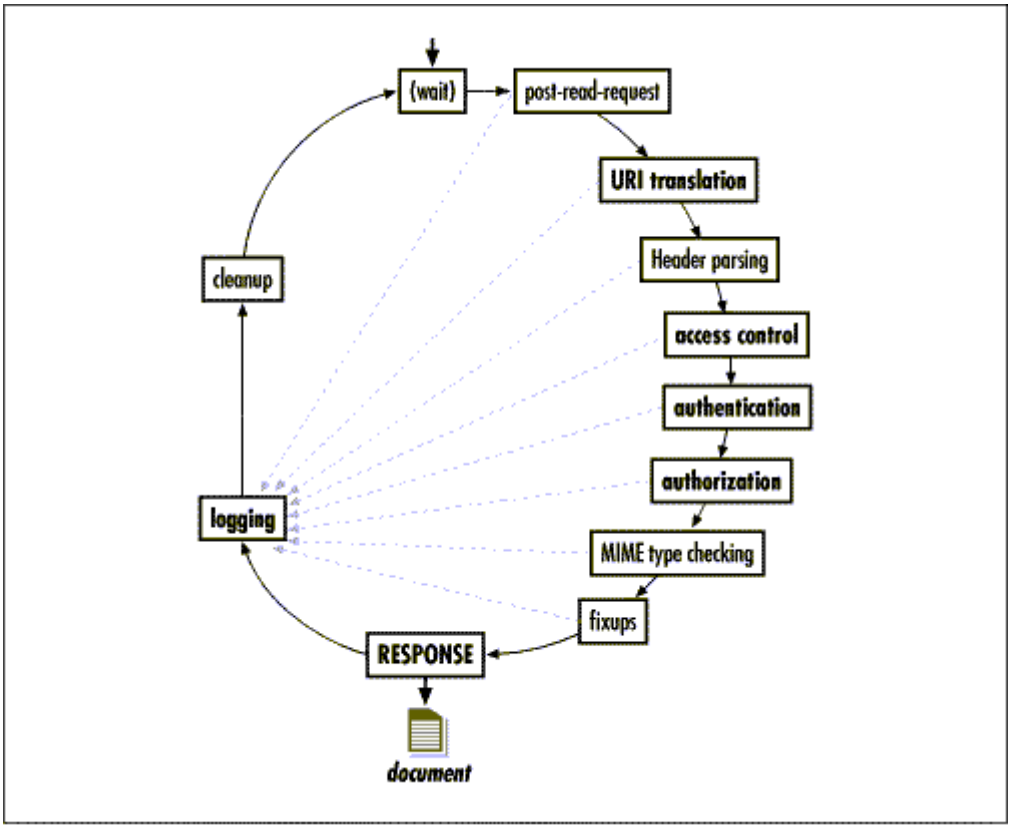
Una volta avviato il **processo** *root*, questo legge la sua configurazione e quindi si mette in ascolto di connessioni ***client***, tipicamente sulla porta 80. Il ciclo di funzionamento di *apache*, per altro identico a quello di ogni altro *Web* ***server***, può essere riassunto nei punti seguenti:

- Una volta avviato, il ***server*** è un demone in stato di attesa (*listening*) di una connessione ***client***.
- La richiesta di un ***client*** apre una connessione col ***server***, il quale è in grado di soddisfare soltanto quella. Per soddisfare le successive il demone lancia un **processo** figlio a cui fa gestire la ***socket*** *TCP*.
- Il demone è quindi ancora attivo e disponibile a stabilire nuove connessioni.



Ciclo di vita del processo

Il vantaggio di una struttura a processi è notevole: il ***server*** è snello (il demone HTTPd occupa poche decine di KB) poiché deve essere in grado di risolvere una sola richiesta alla volta. L'interruzione (o il blocco improvviso) di qualunque **processo** figlio non manda in *tilt* il ***server***, poiché il padre è in grado di continuare a funzionare in maniera indipendente da ogni suo **processo** figlio. Il ciclo di richiesta/risposta che realizza il *Web* ***server*** è illustrato nella figura seguente.

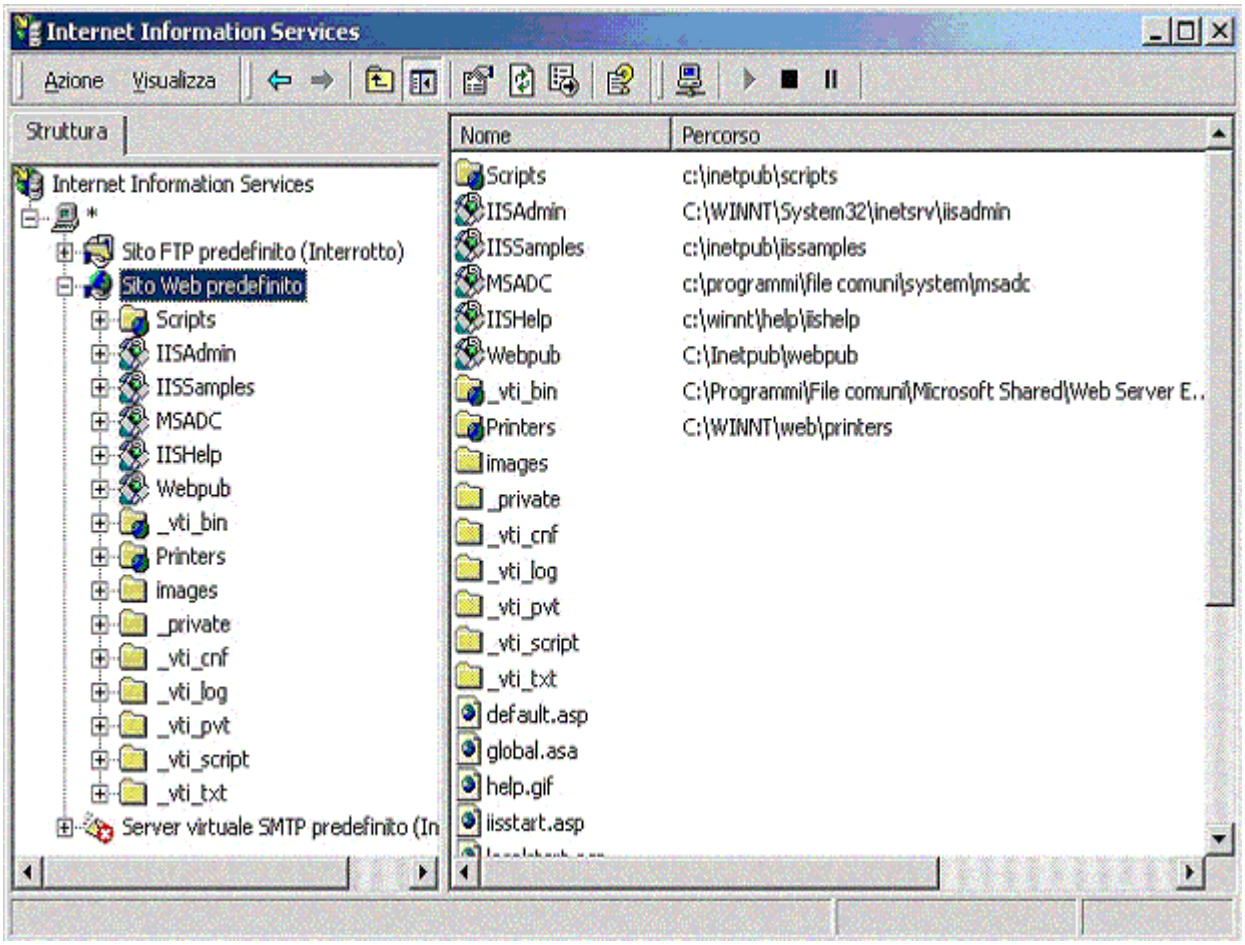


Ricordiamo inoltre che ***Apache*** è distribuito come *free software*, per esplicito desiderio del *team* che lo sviluppa: questo, infatti, ritiene che strumenti di questo genere debbano essere accessibili a tutti, e

che le *software house* debbano guadagnarci solo producendo *addons* o gestendo servizi, magari personalizzandoli per alcune categorie di utenti. Inoltre, lasciando il *software* libero e completo di sorgenti, è più facile controllarne gli errori di sviluppo tramite *feed back*, data la semplicità con la quale si accede al sistema.

Server Web: Microsoft Internet Information Server

Windows 2000 ***server*** comprende un *Web Server* che va sotto la denominazione di *Internet Information Services* (**IIS**). **IIS** viene installato di *default* come servizio di rete durante l'installazione di *Windows* 2000 e permette di supportare anche configurazioni di *Web Server* abbastanza complesse (*Server Web* Virtuali, Cartelle Virtuali, ...). *Internet Information Server* (**IIS**) di *Microsoft* consente di creare applicazioni che fanno uso di pagine *Web* dinamiche per utilizzo pubblico su ***Internet*** o all'interno di reti aziendali. È una piattaforma per applicazioni ***Internet****/Intranet*, utilizzabile soltanto in ambiente *Windows* NT, o superiori. Le applicazioni sviluppate sono compatibili con qualsiasi *browser Web*, funzionante con qualsiasi sistema operativo su ***client***.



IIS include un efficiente motore **HTTP**, servizi **FTP**, SMTP e *Gopher*. Esso è costituito da una serie di pacchetti applicativi, che offrono un ambiente per sviluppare applicazioni, un motore di ricerca, un supporto per trattare elementi multimediali. I pacchetti più importanti sono:

- ***Active Server Page*****:** permette di creare rapidamente potenti applicazioni *Web* attraverso l'integrazione di elementi **HTML**, ***script*** e *component*. È possibile inserire moduli *Visual Basic* e *JScript*, mantenendo la compatibilità con qualsiasi motore di *scripting*, come *Perl*, e gli altri linguaggi ***CGI***. *Active Server Pages* incorpora inoltre una funzione avanzata di accesso via *Web* a *database* aziendali.
- ***Microsoft NetShow*****:** è una piattaforma *software* aperta per la trasmissione, sia in tempo reale che su richiesta, di contenuti multimediali su ***Internet*** e sulle *Intranet* aziendali. Per le

trasmissioni in tempo reale, usa la diffusione selettiva (*multicasting*) per distribuire simultaneamente *file* audio e dati; per quanto riguarda invece la trasmissione su richiesta, permette di memorizzare e inviare contenuti audio, video e *Illustrated Audio* (audio sincronizzato con immagini, indirizzi **URL** e ***script***). Inoltre, *NetShow* supporta l'*ActiveMovie Streaming Format*, che assicura avanzate funzionalità multimediali per *authoring* e sincronizzazione.

- ***Microsoft Index Server*:** è un motore di ricerca integrato, in grado di indicizzare automaticamente testi completi e proprietà dei vari *file*, compresi quelli in formato **HTML**. *Index Server* riesce inoltre a reperire tutti i documenti indipendentemente dal tipo di formato, che può essere un documento *Microsoft Word*, un foglio elettronico *Microsoft Excel*, o una pagina **HTML**. L'indice viene dinamicamente aggiornato quando i documenti cambiano, mentre le funzionalità di sicurezza sono strettamente integrate con quelle di *Microsoft Windows NT*.
- ***FrontPage Server*:** permette la creazione e la gestione delle pagine di un sito ***Internet***/*Intranet*. È uno strumento *client/server* visuale facile e potente per sviluppare e mantenere aggiornati siti *Web*. L'interfaccia utente è coerente con quella di *Microsoft Office* pertanto è facile da usare. La sua architettura *client/server* consente la creazione di pagine *Web*, la preparazione di ***script*** di comandi e la gestione di un sito *Web* anche da una macchina remota.

I benchmark sui server Web

Diversi produttori e riviste si sono avventurati in confronti sui prodotti di gestione dei ***server*** *Web*, creando non poca confusione. Questi confronti, detti ***benchmark***, sono test che intendono mostrare le *performance* di un prodotto con *software* e *hardware*.

I due *Web* ***server*** maggiormente diffusi sulla rete sono, senza dubbio alcuno, *Microsoft Internet Information Server* (**IIS**) e ***Apache***. La differenza principale fra i due applicativi è la piattaforma per la quale sono stati pensati: **IIS** è per ***server*** basati su sistemi *Windows*, ***Apache*** per ***server*** basati su *Unix*.

Di seguito sono discussi alcuni ***benchmark*** effettuati utilizzando macchine simili con configurazioni *Linux/**Apache*** e *Windows* NT/**IIS**.

Il primo test di confronto tra ***Apache*** e *Microsoft* **IIS** è stato effettuato facendo richiedere a tutti i ***client*** la stessa pagina **HTML** da 4Kb: il risultato è una sostanziale parità di *performance* fra i due *Web* ***server***.

Il secondo, invece, si basa su una richiesta casuale da parte dei ***client*** di una pagina fra le 10.000 (ovviamente della stessa grandezza) presenti in una *directory* del ***server***. In questo test, *Linux/**Apache*** è risultato circa il 15% più veloce.

Il terzo test, che inizia ad essere impegnativo per le macchine, si basa sulla richiesta di una quantità di *file* di due volte superiore a quella della RAM dei ***server*** (quindi qualcosa come 4 GB di pagine richieste). NT/**IIS** sembra non riuscire a sopportare più di 30 richieste al secondo, mentre *Linux/**Apache*** arriva a 166. La differenza, si commenta, può essere principalmente dovuta al tipo di partizione utilizzata dai sistemi (in termine di grandezza dei *cluster* o *i-node*) che non alle capacità degli stessi.

Il quarto test si basa non più su pagine statiche come nei precedenti tre test, ma su pagine dinamiche: la scelta è ricaduta sui ***CGI***, non essendo le tecnologie **ASP** e *VBscript* direttamente portabili ad altri sistemi. Su NT è stato installato *ApcivePerl* 517.3, equivalente al *Perl* 5.005_3 presente sul ***server*** *Linux*.

Il risultato è palese: NT/**IIS** soffrono della non-natività del linguaggio *Perl* utilizzato per l'interpretazione dei ***CGI***, riscontrabile come un'eccessiva lentezza nell'invio delle pagine create dinamicamente. Ma per questo test, avvertono, il risultato era scontato dall'inizio, e quindi poco significativo. Ricordiamo infatti che il *Perl*, linguaggio più largamente utilizzato per scrivere *script* ***CGI***, è nato in ambiente *Unix* e, sebbene il *porting* di *Activestate* per le piattaforme *Windows* sembra essere molto ben riuscito, ancora una volta la non-natività del linguaggio di interpretazione si fa sentire.

L'ultimo test è quello di servire sedici macchine tramite due schede di rete. Qui NT/**IIS** riesce a prevalere sull'avversario *Linux/**Apache*** anche con una potenza di calcolo minore, ossia con un processore in meno. La velocità con cui NT/**IIS** invia le pagine ai ***client*** ha fatto addirittura pensare che le *performance* non sarebbero peggiorate significativamente neanche con quattro schede di rete al posto di due.

Le conclusioni degli autori del test sono le seguenti: per una rete mista, formata cioé da richieste differenti ed indipendentemente dal volume delle stesse, l'accoppiata *Linux/**Apache*** risulta migliore. Per un ***server*** con più schede di rete e con prestazioni medio-alte, NT/**IIS** è l'accoppiata ideale. Ovviamente, le necessità in pratica possono essere anche altre, ad esempio il **linguaggio di scripting** da adottare (***CGI***, **ASP**, *VBscript*, eccetera), la potenza della macchina da utilizzare (sembra infatti che *Linux/**Apache*** riesca a spremere maggiormente le *CPU*, sebbene NT/**IIS** appaia sempre più performante per ogni *CPU* aggiunta), eccetera.

Installazione e configurazione di un Web server

Il **processo** di creazione di un *Web Server* è un **processo** molto simile a quello relativo alla creazione di un *file* ***server***. Come nel caso di ogni altra tipologia di ***server*** bisogna innanzitutto installare il sistema operativo, decidere l'appartenenza ad un gruppo di lavoro (*stand-alone* ***server***) o ad un dominio (*member* ***server***), ed organizzare in maniera opportuna i dischi. Il passo successivo consiste nell'installare e configurare in maniera appropriata i servizi relativi alla funzionalità di *Web Server*. Ricordiamo che tramite un *Web Server* un ***client*** accede a *file* che costituiscono le pagine *Web*, ma anche a complesse applicazioni *client/server* basate su *database*.

Brevemente, per configurare un *Web **server*** sono necessari:

- Installare un *TCP/IP*.
- Un **indirizzo** *IP* statico.
- Un nome di domino.

Installazione e configurazione di Apache

Ci sono due possibilità di installazione per ***Apache***: il nuovo **processo**, detto APACI, è più veloce e comodo, poiché utilizza tutte le modalità standard di installazione dei pacchetti sotto *Linux*; il **processo** più vecchio è invece quello manuale, lungo e noioso. Vediamo come si svolge il tutto per il primo:

- dobbiamo innanzitutto assicurarci di avere risorse disponibili: dobbiamo avere circa 12 Mb di spazio temporaneo su disco e 3 Mb necessari all'installazione; serve poi il compilatore, che presumibilmente avrete già installato: è vivamente consigliato GCC 2.7.2 o superiore. Vengono poi citati come opzionali l'interprete *perl* 5 (che comunque dovrete avere, altrimenti i ***CGI*** saranno impossibili da eseguire) ed il supporto per i DSO (*Dynamic Shared Object*, che servono per alcune chiamate di sistema per il caricamento dei moduli esterni);
- lanciamo lo ***script*** `./configure`, per preparare la compilazione dei sorgenti; tale ***script*** accetta varie opzioni:
    - `--prefix=PREFIX`. È la *directory* nella quale volete installare ***Apache*** (ovviamente

PREFIX dovrà essere cambiato con il nome della *directory*): consigliata `/usr/local`
    - `--add-module=FILE`. Serve per copiare il sorgente di un modulo nell'albero dei sorgenti di ***Apache***; ovviamente al posto di *FILE* dovrete inserire il *path* per il sorgente del modulo.
    - `--activate-module=FILE`. Aggiunge al volo un'*entry* per un modulo al *file* di configurazione di ***Apache***.
    - `--enable-module=NAME` . Serve per fare in modo di abilitare (o disabilitare, utilizzando *--disable-module*) un modulo particolare.
    - `--with-perl=FILE` . Serve per impostare l'interprete perl utilizzato da ***Apache*** (anche se di solito ***Apache*** cerca da solo l'interprete).
- Ovviamente la opzioni da passare a `./configure` non sono tutte qui: si è pensato di presentarvi solamente le più utili (soprattutto al nostro scopo). Per conoscere tutte le opzioni, lanciare `'./configure --help'`.
- Iniziamo la compilazione lanciando '`make`'.
- In seguito si installa: *make install* è il comando da lanciare.
- A questo punto ***Apache*** è installato ma non ancora attivo: lanciamo `PREFIX/sbin/apachectl start` e dovremmo poter richiedere ad ***Apache*** il primo **URL**, che sarà http://localhost. Se volete fermare ***Apache***, lanciare `PREFIX/sbin/apachectl stop`. Per la configurazione il *file* principale è `/etc/apache/HTTPd.conf`, ma è possibile averlo da qualche altra parte, soprattutto se si è partiti compilando i sorgenti.

Configurazione di Apache

Vediamo com'é strutturato *HTTPd.conf*: intanto, ogni voce è ampiamente commentata, così da farvi capire cosa state facendo e a cosa serve quello che state facendo. Le voci più significative sono:

Voce	Descrizione
ServerType	Il tipo di ***server***, '*standalone*', è la impostazione *standard*
Port	È la porta. Di *default* ha valore 80
HostnameLookups	Logga i nomi dei ***client*** (*on*) o solamente il loro numero *IP* (*off*); lasciatelo pure su *off*
ServerAdmin	Se si imposta un indirizzo *e-mail*, su ogni errore, verrà incluso nei messaggi inviati ai ***client***.
ServerRoot	La *directory* dove ***Apache*** conserva i *log*, gli errori e i *file* di configurazione; solitamente è la *directory* dove è presente anche il *file* `HTTPd.conf`
LoadModule	Indica ad ***Apache*** quali moduli caricare. Indispensabili sono tutti i moduli per i ***CGI***, per il *Perl* e i *mime* (commentati)
ErrorLog	Il *file* dove ***Apache*** scrive gli errori. `/var/log/apache/error.log` è quello di *default*
ServerName	Il nome del vostro ***server***. Se non impostato, sarà `localhost.localdomain` ma, siccome è assai scomodo, lanciate (da *root*) *hostname* e cambiate il nome al vostro ***host***

Un altro importante *file* da configurare è `srm.conf`, che dovrebbe essere nella stessa *directory* di `HTTPd.conf`.

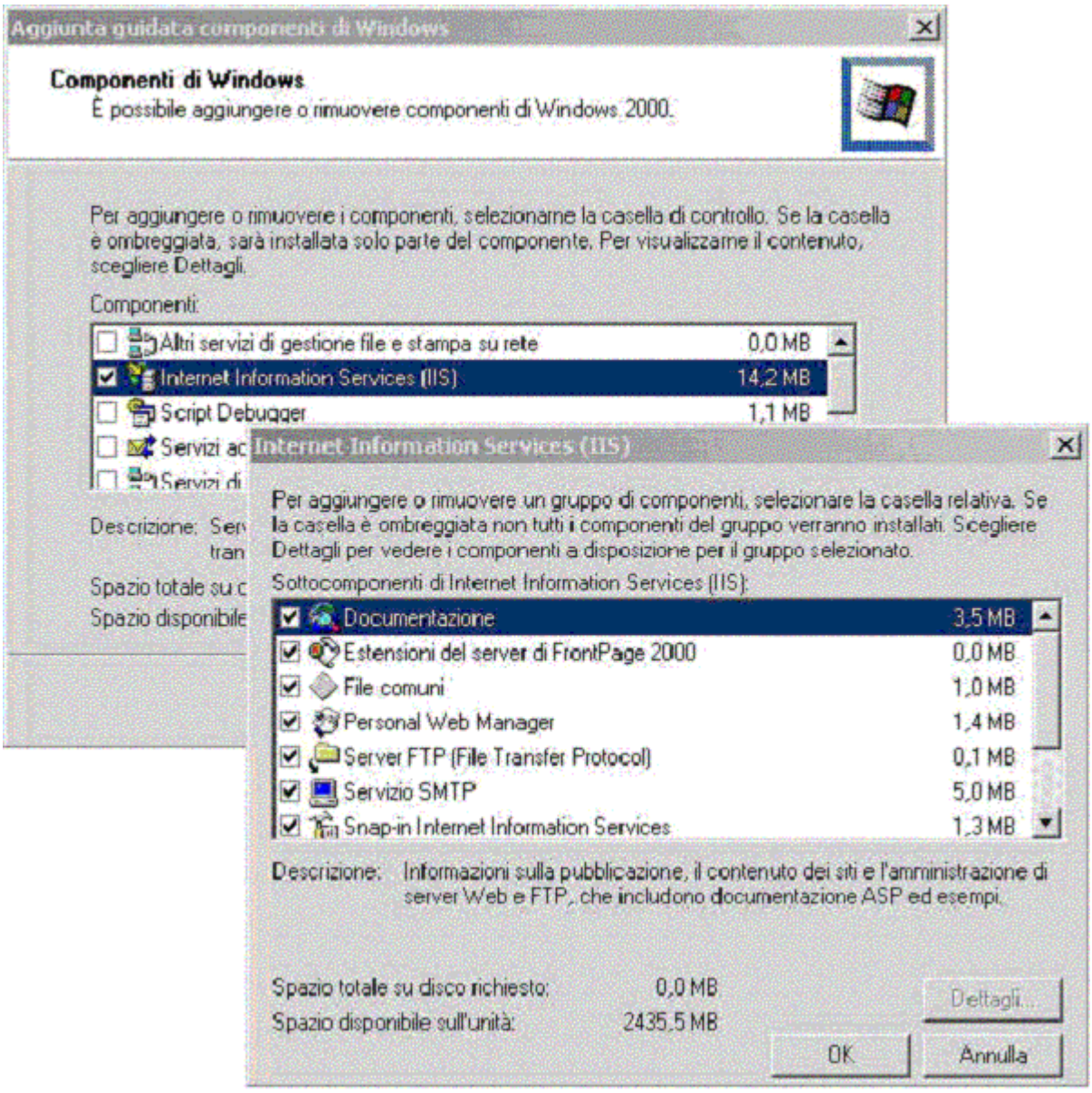
Voce	Descrizione
DocumentRoot	La *directory* nella quale mettere i *file* html per la pagina locale. Solitamente `/var/www`: quindi se in tale *directory* mettete un vostro *index.html*,

	indicando '*localhost*' come **indirizzo** del *browser*, vedrete proprio questa pagina. E da questa partiranno le altre pagine
DirectoryIndex	Il nome della pagina che verrà visualizzata come indice; solitamente *index.html*
ScriptAlias	Ed eccoci arrivati ai ***CGI***: conviene impostare tale voce come: `ScriptAlias /cgi-bin/ /usr/lib/cgi-bin/`. Cosa significa? Semplicemente, che voi potrete inserire i vostri ***CGI*** in `/usr/lib/cgi-bin/`, ma questi saranno chiamati tramite http://localhost/cgi-bin/nome_cgi.cgi

Se qualcosa non dovesse funzionare al meglio, leggete la documentazione inclusa nell'archivio che avete scaricato ed eventualmente cercate la cartella `/usr/doc/apache`.

Installazione e configurazione di Web server: Microsoft IIS

Fare clic sul pulsante *Start*, scegliere Impostazioni, quindi Pannello di controllo. Fare doppio clic sull'icona Installazione applicazioni, scegliere Configurazione di *Windows*, quindi Componenti. Seguire le istruzioni visualizzate per installare, rimuovere o aggiungere componenti di **IIS**.



Come installare *Visual Interdev*

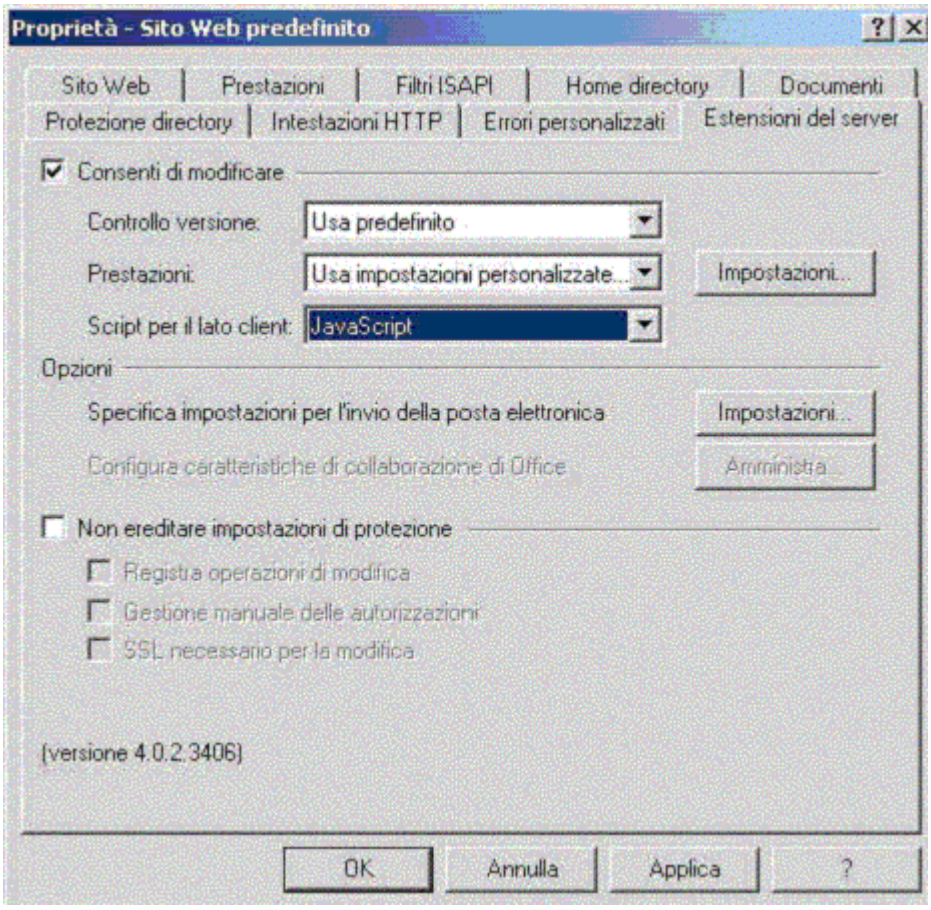
Se si desidera disporre di un ***server*** locale, installare **IIS** servendosi dello strumento Installazione applicazioni prima di installare *Visual InterDev*. Se si installa *Visual InterDev* 6.0 in un *computer* in cui è installato *Visual InterDev* 1.0, è necessario disinstallare la versione 1.0 prima di installare la versione 6.0. Se sono già stati installati componenti di *Visual* Studio prima di installare *Visual InterDev*, è necessario avviare l'installazione di *Visual InterDev* facendo clic su *Microsoft Visual* Studio versione del prodotto nello strumento Installazione applicazioni del Pannello di controllo e scegliendo Aggiungi/Rimuovi.

Dopo avere installato il ***client*** di *Visual InterDev* sul *computer*, viene richiesto di installare *Microsoft Developer Network* (MSDN). È necessario installare MSDN perché il tasto F1 della Guida funzioni e si possa accedere alla documentazione di *Visual InterDev*. Se MSDN è già stato installato e non si desidera installarlo, deselezionare la casella di controllo Installa MSDN, quindi scegliere Fine.

Installare le estensioni del server

Per installare le estensioni del ***server*** di *FrontPage* 2000 per *Microsoft Internet Information Server*, fare doppio clic su *file* eseguibile scaricato dal sito *Web Microsoft* e seguire le richieste visualizzate. Le estensioni del ***server*** di *FrontPage* verranno configurate esclusivamente per il sito *Web* predefinito. È possibile effettuare l'istallazione soltanto utilizzando l'*account* di amministratore o un *account* del gruppo *Administrators*. È possibile scaricare le estensioni del ***server*** di *FrontPage* 2000 dal sito *Web Microsoft*: http://msdn.microsoft.com/workshop/languages/fp/2000/winfpse.asp.

Scaricare ed installare il *file Fpse2k_x86_eng.exe* (circa 14 MB). È necessario effettuare un riavvio del *computer*, in seguito al quale verranno eseguite altre operazioni di installazione. Fare doppio clic su Strumenti di amministrazione, quindi fare doppio clic su Gestione servizio *Internet Microsoft*. Fare clic sul segno più + accanto al nome del ***server*** per espandere il ramo. Fare clic con il pulsante destro del *mouse* su Sito *Web* predefinito, scegliere Tutte le attività, quindi configurare le estensioni del ***server***. Nella Configurazione guidata estensioni del ***server***, scegliere Avanti.

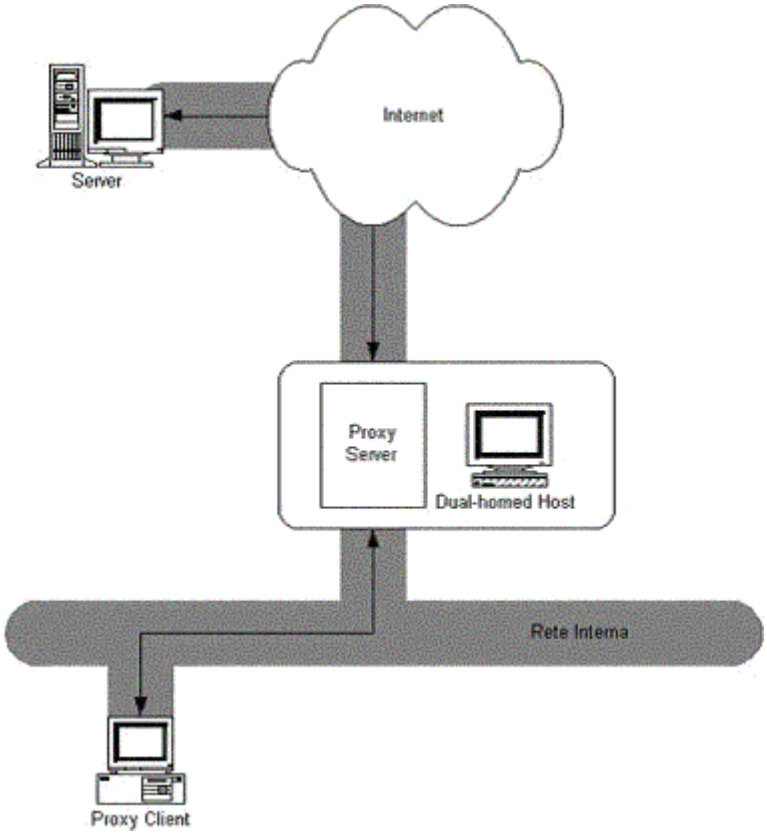


Attenzione: scegliere No se viene visualizzato il seguente messaggio: Se si mantiene la configurazione corrente del ***server*** *Web*, la modifica delle pagine disponibili nel ***server*** dopo l'installazione delle estensioni sarà consentita a chiunque. Continuare con l'installazione delle estensioni?

A questo punto è necessario configurare la partizione in cui è residente **IIS** in modo che utilizzi il *file system* NTFS anziché FAT o FAT32 (da evitare accuratamente soprattutto sulle macchine di produzione). Per risolvere questo problema sospendere l'installazione delle estensioni e riconfigurare **IIS** in modo che utilizzi una partizione NTFS. Configurare le impostazioni del ***server*** di posta digitando il nome del ***server*** SMTP, quindi scegliere Avanti. Si noti che è necessario effettuare questa configurazione solo se dal sito *Web* verranno inviati messaggi di posta elettronica, mentre non è necessaria per la maggior parte delle funzionalità **ASP**.

Descrizione dei proxy server, installazione e configurazione - I proxy server

Un ***proxy*** *server* è un *computer* che funge da interfaccia nelle connessioni a ***Internet*** tra l'utente e ***Internet*** stessa, fornendo importanti funzionalità di rete.



I ***proxy*** *server* forniscono essenzialmente quattro funzionalità:

- *Firewall* e filtraggio dei messaggi.
- Condivisione della connessione tra più utenti.
- Memorizzazioni delle pagine in memoria *cache*.
- *Gateway* per connettere la rete locale a ***Internet***.

I ***firewall***, come vedremo nel **capitolo successivo** dell'introduzione, consentono il passaggio di messaggi in entrata solamente a repliche di messaggi in uscita precedenti, per evitare di comprometterne la sicurezza. I filtri invece consentono ad un amministratore di disabilitare l'accesso a particolari domini. La condivisione della connessione tra utenti consente l'utilizzo della stessa connessione ad ***Internet*** da parte di più utenti. Il *caching* consente di migliorare la qualità del

servizio perché non spreca la banda di comunicazione della rete, produce risposte più veloci, e permette l'accesso a pagine *Web* anche quando il sito corrispondente è inattivo. Infine, per il servizio di *gateway* della rete locale a ***Internet***, facciamo un esempio. Supponete che il vostro istituto abbia sottoscritto degli abbonamenti a pagamento presso riviste. Naturalmente questi siti sono accessibili dal *computer* sulla vostra scrivania in ufficio. Da casa invece, in cui tipicamente siete nel dominio dell'*Internet Provider*, non c'è possibilità di accedervi. Se non settando opportuni parametri del vostro *browser* per accedere al ***proxy*** *server* dell'istituto (quando questo lo consente!).

Proxy Server e browser

I ***proxy*** *server* operano con specifici protocolli di rete, di cui **HTTP** è il più importante ed il più critico perché consente di accedere a pagine *Web*. Questi protocolli sono:

- **s-HTTP**, o *secure*-HTTP, che consente comunicazioni **HTTP** criptate e sta diventando sempre più comune soprattutto nei siti di commercio elettronico per espletare transazioni economiche (che prevedono il numero di carta di credito, per esempio). Non va confuso con **ssl**, che è un **protocollo** più a basso livello utilizzato da **s-HTTP** ma che non ha niente a che fare con il *server* ***proxy***.
- **FTP** (*File Transfer Protocol*), che consente il *download* o l'*upload* di *file* dalla/alla rete. Il protocollo **FTP** tratta *file* in formato testo o in formato binario ed è tuttora molto utilizzato per scaricare archivi compressi, come i *file* MP3.
- ***socks***, implementa ***firewall*** di sicurezza e viene usato per *chat* e *telnet*.
- ***gopher*** e ***wais***, sono due protocolli poco usati: sono stati due tentativi di standardizzazione prima di **HTTP**.

Tecnicamente si potrebbero utilizzare ***proxy*** *server* differenti per far fronte ai diversi protocolli. Ma questo significherebbe che i clienti che usano i ***proxy*** *server* devono essere a conoscenza di dettagli di comunicazione. Sicché, gli amministratori quasi sempre configurano i ***proxy*** in modo da operare con tutti i protocolli.

Installare e configurare un proxy server

Sono necessarie due informazioni per specificare manualmente un ***proxy*** *server* nel *browser*:

- il **nome dell'*host*** (come configurato nel **DNS**), oppure il suo numero *IP*;
- il **numero di porta *TCP/IP*** su cui il ***proxy*** *server* sarè in attesa di richieste.

Di solito la porta è la stessa per tutti i protocolli di prima (**HTTP**, **s-HTTP**, **FTP**, eccetera), ed è differente dal numero di porta assegnata ai vari protocolli (spesso 80 per **HTTP**, 21 per **FTP**, eccetera).

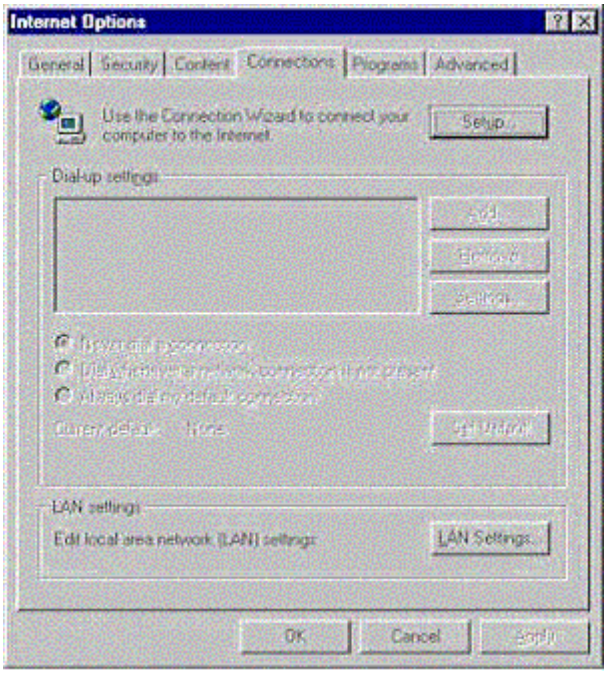
Recentemente, per agevolare gli amministratori nella configurazione dei ***proxy*** *server*, sono stati messi a punto alcuni sistemi. Gli amministratori possono usare opportuni *file* di configurazione con codice *JavaScript* per nascondere agli utenti dettagli quali i numeri di porta. Gli utenti dovranno semplicemente accedere via *browser* a questi *file* per configurare tutti i parametri.

Microsoft ha messo a punto una nuova tecnologia per *Explorer* (dalla versione 5 in poi), detta *Web Proxy Auto Discovery* (WPAD), per scoprire la presenza di *server* ***proxy*** e di altri servizi *Web*. WPAD usa un servizio di ricerca come **DNS** per creare un *file* di configurazione che gli amministratori possono installare nel *Web* ***server***. Gli amministratori debbono soltanto inserire in questo *file* gli indirizzi dei servizi *Web* relativi.

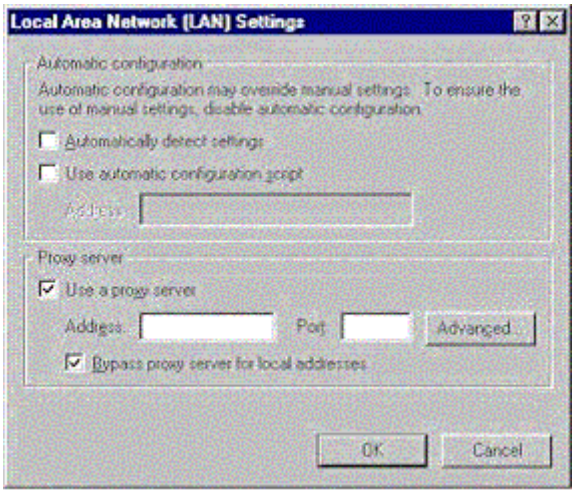
Proxy server e Microsoft Internet Explorer

Per configurare il *Proxy Server* in *Internet Explorer* occorre accedere a *Tool* nel menù di *Internet Explorer*.

Quindi bisogna selezionare *Internet Options*. A questo punto appare una finestra con diversi fogli, relativi a differenti funzionalitè. Selezionare il foglio delle connessioni di rete (*Connections*).



A questo punto bisogna selezionare il bottone in basso sui parametri della rete locale (***LAN** settings*). Quindi compare una finestra come la seguente:



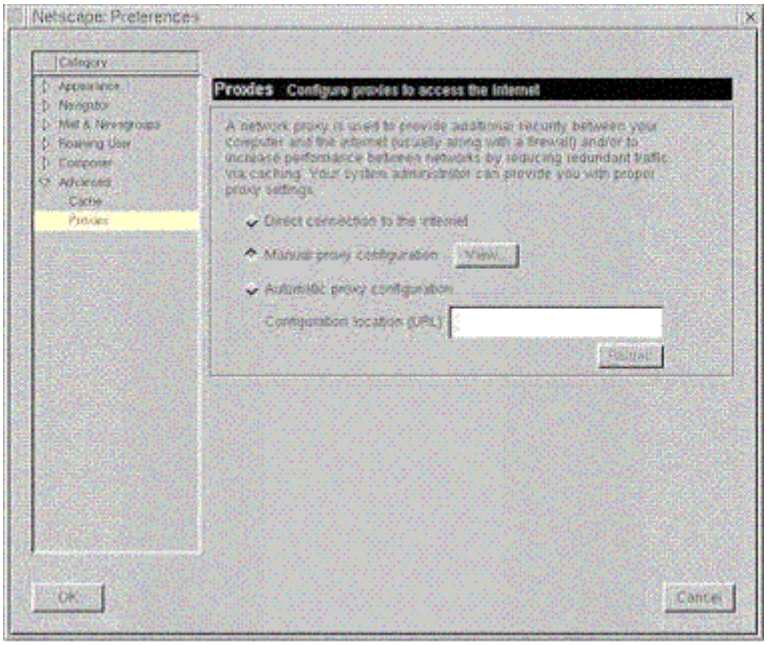
In *Internet Explorer* è possibile sia la configurazione automatica che manuale. Per quella manuale

bisogna selezionare, come mostrato nella figura di sopra, l'opzione *Use a **proxy** server*. Quindi nel campo di **indirizzo** bisogna inserire il nome dell'***host*** oppure l'**indirizzo** *IP* del ***proxy***. Nella configurazione manuale, bisogna selezionare, nella stessa finestra di sopra, l'opzione *Automatically detect settings* che utilizza il meccanismo WPAD per scoprire la configurazione del ***proxy***. Infine, selezionando *Use authomatic configuration script*, è possibile specificare l'**indirizzo URL** del *file* di configurazione di *JavaScript*.

Proxy server e Netscape Navigator

In *Netscape* si inizia selezionando *Edit* nel menù.

Quindi selezionare *Preferences*. A questo punto compare la finestra di configurazione di *Netscape* che è visualizzata sotto. In questa finestra bisogna selezionare *Advanced* e poi *Proxies*.



Anche *Netscape* consente la duplice configurazione automatica e manuale di ***proxy** server*. Infatti, a questo punto, si può scegliere l'ultima opzione per la configurazione automatica, quella centrale per quella manuale (in tal caso bisogna selezionare *View* per una ulteriore finestra che richiederà il nome del ***proxy** server* o l'**indirizzo** *IP*). Infine, la prima opzione consente di collegarsi a ***Internet*** in modo diretto, senza utilizzare ***proxy** server*.