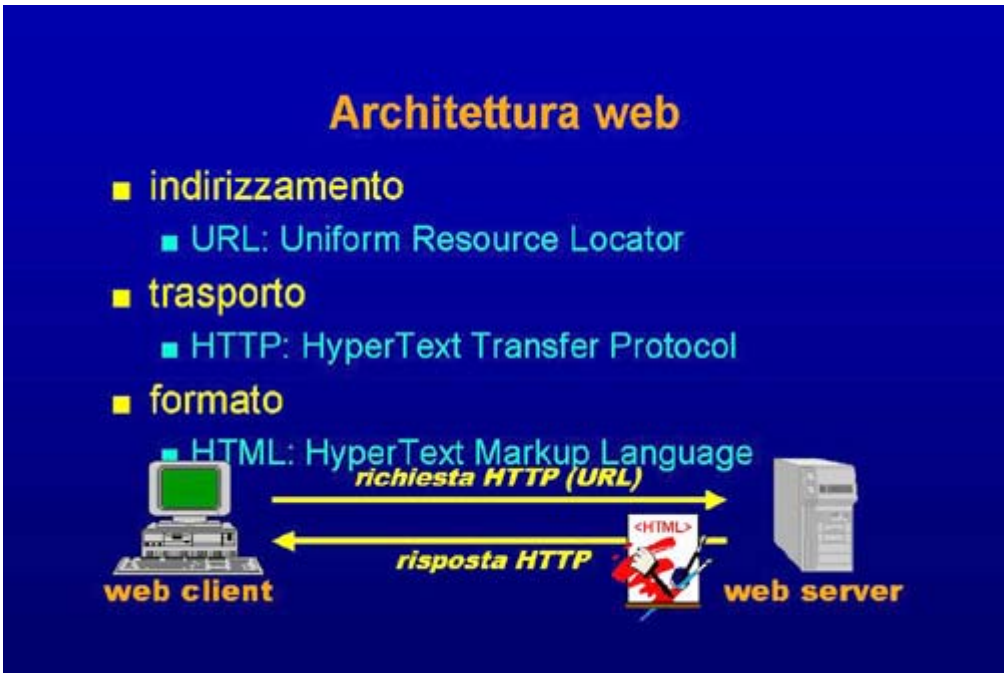
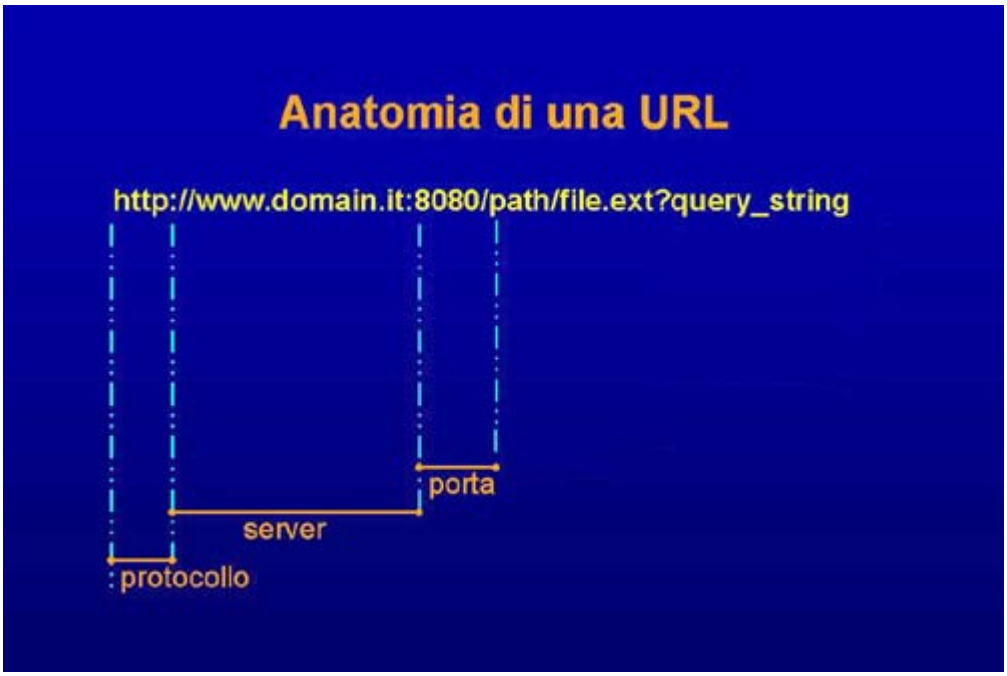
Architetture Web: CGI
Architettura Web



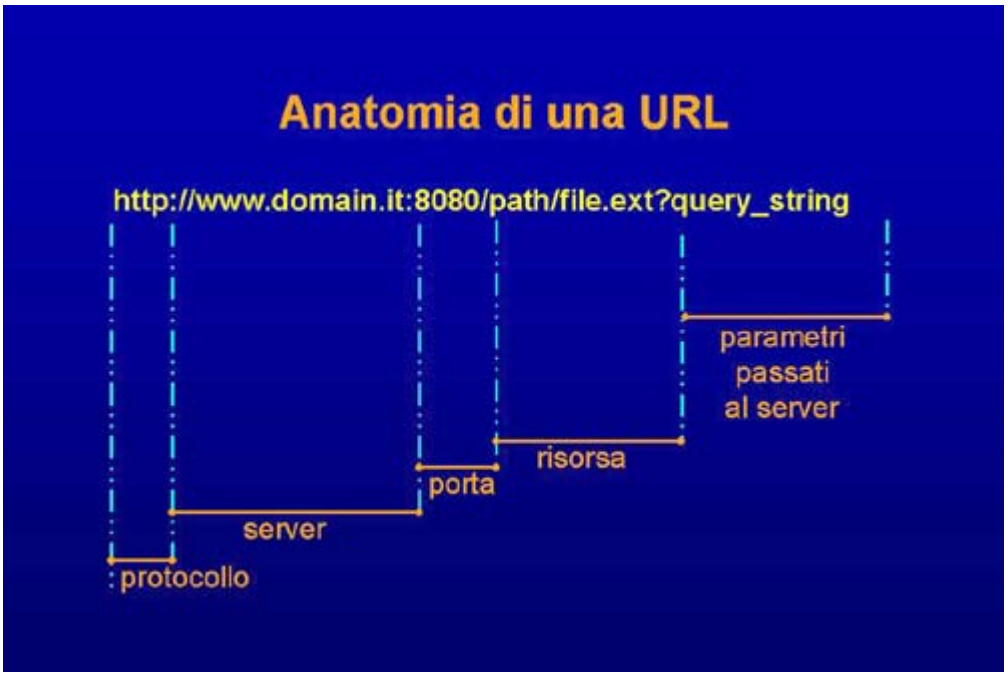
Benvenuti a questa lezione sulle architetture Web. Nella lezione precedente abbiamo appreso quali sono i protocolli utilizzati in Internet. Nel corso di questa lezione e delle successive ci occuperemo, in particolare, del protocollo HTTP e del protocollo CGI. Andremo a vedere quali sono le architetture Web a tre livelli, andando ad analizzare in particolare l'architettura Server Side Include, le architetture Web API e l'architettura a codice mobile. Ci occuperemo, quindi, di quelle che sono le architetture Microsoft per il Web, in particolare la tecnologia ASP e la tecnologia ActiveX. Infine, andremo a vedere quali sono le applicazioni Java che consentono di utilizzare codice mobile su Internet. Cominciamo questa lezione sulla architettura Web. Dovremmo andare a comprendere quali sono i meccanismi di indirizzamento, quali sono i meccanismi di trasporto e quali sono i meccanismi di formato utilizzati nel Web. In particolare, vedremo, che nel Web possiamo utilizzare un Uniform Resource Locator per indirizzare univocamente ciascuna risorsa. Individuata la risorsa che vogliamo indirizzare utilizziamo come protocollo di trasporto: il protocollo HTTP. HTTP sta per HyperText Transfer Protocol ed è un protocollo utilizzato per il trasferimento di ipertesti. Infine, nel corso di questa lezione, vedremo come l'HTML possa essere utilizzato per passare parametri dal client al server Web. Il protocollo HTTP, come indicato in questa slide, prevede essenzialmente due momenti: una richiesta HTTP, con la quale viene invocata una risorsa di cui è stata specificata la corrispondente URL, e una risposta HTTP, con cui il server Web invia il documento richiesto. In questo caso si tratta di una pagina HTML che contiene del testo e delle parti grafiche.

Anatomia di una URL (1/2)



Per comprendere il funzionamento dell'architettura Web dobbiamo comprendere quali sono i meccanismi di indirizzamento. In particolare dobbiamo comprendere qual è l'anatomia di una URL, ovvero quali sono gli elementi che costituiscono una URL e qual è il contenuto d'informazione di ciascuno di questi elementi. Una URL è costituita da una serie di elementi che individuano diverse parti della risorsa che vogliamo andare ad individuare. In particolare, la prima parte della URL individua quello che è il protocollo utilizzato per il trasporto. L'URL è un meccanismo di indirizzamento globale che consente, quindi, di utilizzare per il trasporto delle risorse che vogliamo trasportare, non solo il protocollo HTTP, ma anche altri protocolli, per esempio il protocollo FTP o protocolli di questo tipo. In questo caso, questa URL specifica che la risorsa alla quale siamo interessati è accessibile tramite il protocollo HTTP. La seconda parte di una URL individua qual è il server sul quale si trova la risorsa alla quale siamo interessati, in questo caso il server è quello caratterizzato dall'indirizzo www.domain.it. Il terzo elemento individuato da una URL specifica qual è la porta alla quale è accessibile quel servizio. Normalmente questa parte di URL non è presente, in quanto i server Web rispondono alla porta 80. In questo caso non è necessario indicare la porta specifica alla quale viene richiesta questa risorsa.

Anatomia di una URL (2/2)



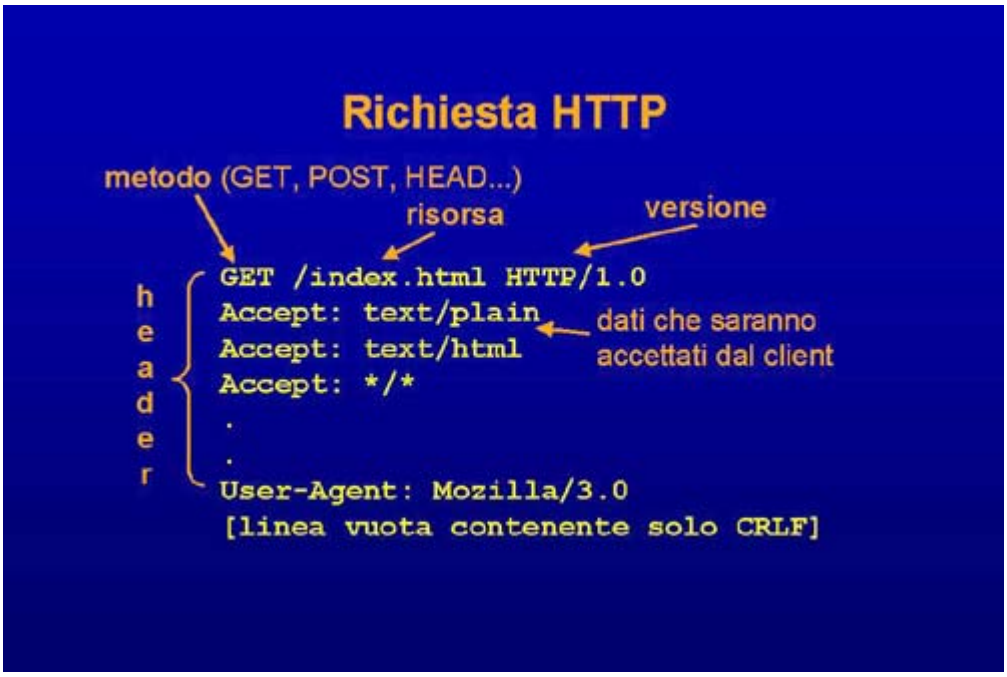
Individuato qual è il server e qual è la porta alla quale è distribuita la nostra risorsa, dobbiamo specificare qual è la risorsa alla quale siamo interessati. Questo può essere fatto attraverso il path name, che individua in maniera univoca un file o un oggetto distribuito da quel server Web. Queste quattro parti individuano la risorsa che vogliamo andare a recuperare su di un server Web. Vedremo che una URL può essere utilizzata non solo per individuare una particolare risorsa, ma anche per passare parametri dal client al server Web. In questo caso viene utilizzata una parte della URL, che va sotto il nome di query_string. Nel proseguo vedremo che questo elemento, la parte query_string di una URL, è utilizzato per inviare dati dal client verso il server.

Il protocollo HTTP

Andiamo a comprendere, ora, quali sono gli elementi che costituiscono il protocollo HTTP, cercando di comprendere appieno quali sono le varie fasi in cui può essere scomposta una connessione HTTP. Il protocollo HTTP è un protocollo client/server che prevede quattro fasi: la

prima è l'apertura della connessione, con la quale il client instaura una connessione diretta verso il server. Aperta una connessione il client invierà una richiesta HTTP, andando a richiedere una ben precisa risorsa. A questa richiesta seguirà la risposta HTTP, con cui il server invia la risorsa che è stata richiesta dal client. L'ultima fase della connessione HTTP è la chiusura della connessione stessa, che normalmente viene eseguita dal server, ma in alcuni casi potrebbe anche essere eseguita dal client, ad esempio perché si è atteso troppo tempo per la risorsa desiderata. È importante notare che HTTP è un protocollo stateless, un protocollo nel quale non esiste nessuna relazione tra due richieste, anche se sono sequenziali e anche se si riferiscono ad oggetti che sono contenuti nella stessa pagina HTML. Sarà il client a tenere traccia delle relazioni che ci sono tra diverse connessioni HTTP. In particolare, se il client intende scaricare una pagina HTML che contiene, oltre a del testo, anche un paio di immagini, ad esempio, sarà il client che dovrà eseguire due ulteriori connessioni verso il server, per scaricare le due immagini contenute nella pagina HTML.

Richiesta HTTP



Andiamo ad analizzare come sono strutturate sintatticamente le richieste e le risposte HTTP. Questo consentirà di capire in che modo client e server possono scambiarsi dati utilizzando il protocollo HTTP e in che modo il client possa passare dei parametri al server. Cominciamo quindi dalla richiesta HTTP. Una richiesta HTTP prevede una serie di elementi e generalmente è composta di due parti, una parte di header, che vediamo indicata qui, separata tramite una linea vuota, che contiene soltanto un carattere di ritorno a capo, dalla parte di body. La parte di body è opzionale e potrebbe non essere presente, come in questo caso. Il primo parametro specificato in una richiesta HTTP è il metodo utilizzato. Si tratta del meccanismo con il quale viene richiesta una determinata risorsa HTTP. Esistono diversi metodi e impareremo a conoscerne le differenze nel corso di questa lezione. Il secondo elemento contenuto in una richiesta HTTP individua qual è la risorsa desiderata. In questo caso stiamo accedendo al file index.html a partire dalla radice del Web server stesso. Il terzo parametro, che ritroviamo sulla prima riga della richiesta HTTP, specifica quale versione del protocollo HTTP stiamo utilizzando; in particolare, in questo caso, stiamo utilizzando la versione 1.0. Il client specifica poi tutta un'altra serie di parametri rivolti al server, con la quale descrive alcune delle sue caratteristiche. In questo caso, per esempio, il client specifica che accetterà file di tipo text in formato plain o in formato HTML e nell'ultima riga, che vedete nell'header HTTP, specifica che si tratta di un client di tipo Mozilla, versione 3.0. Questo consente di passare alcune informazioni sul tipo di client che andiamo ad utilizzare al server che potrà utilizzarle ad esempio per personalizzare l'output per un particolare tipo di server.

Risposta HTTP (1/3)

Andiamo a vedere, ora, in che modo è organizzata la risposta HTTP, ovvero in che modo il server, a seguito di una richiesta proveniente dal client, specifica i dati da trasmettere al client. Così come la richiesta HTTP, anche la risposta HTTP è divisa in due parti. Anche in questo caso la prima parte (header) è separata dalla parte di body da una linea vuota, contenente soltanto il carattere di ritorno a capo. Nella parte di body possiamo riconoscere immediatamente il contenuto di questa risposta HTTP. Si tratta, in questo caso, della pagina HTML che il server sta distribuendo al client. Riconosciamo le tipiche tag che descrivono la struttura di questa pagina HTML. Andiamo ad analizzare però la parte di header della risposta HTTP e cerchiamo di comprendere quali sono le informazioni che il server invia al client. La prima di queste è una informazione di stato: vedete che, in questo caso, il server segnala che la richiesta è stata assolta correttamente. In particolare viene utilizzato un codice numerico, il cosiddetto three digit code, con cui il server specifica lo stato della risposta. La prima cifra di questo three digit code specifica il fatto che la richiesta sia andata a buon fine, oppure che ci sia stato un errore. Le altre due cifre consentono di specificare meglio che tipo di errore si è verificato, o se la richiesta è andata a buon fine. Vi sarà familiare, ad esempio, nel corso delle vostre navigazioni Web, il codice di errore 404 corrispondente ad una pagina HTML non trovata. Questo è il tipico codice di errore restituito da un server Web quando viene richiesta una risorsa non presente su quel server Web. Tutti i messaggi di errore di un server cominciano con il codice 4.

Risposta HTTP (2/3)

Torniamo alla nostra risposta HTTP e andiamo a vedere quali sono gli altri elementi. Il server indica anche quale versione del protocollo HTTP andrà ad utilizzare per la sua risposta. E indica un'altra informazione particolarmente importante. Come notate in questa riga, il server specifica il tipo ed il sottotipo di documento che andrà a spedire. Questa informazione è fondamentale perché il client dovrà comprendere che tipo di documento gli è stato spedito. Non è possibile utilizzare a questo scopo la convenzione, utilizzata localmente dai sistemi operativi, di desumere il tipo di file dall'estensione stessa. Potrebbe succedere, infatti, che su un determinato computer un estensione abbia un significato e abbia un significato diverso su di un altro computer. Ad esempio, molto spesso, nei PC con sistema operativo Windows i file HTML hanno estensione .htm, mentre molto spesso sulle macchine Unix i file HTML hanno estensione .html. Questo fatto renderebbe incomprensibile le due estensioni in due macchine che usano un sistema operativo diverso. Per evitare questo problema, come vediamo in questa slide, il server specifica il tipo ed il sottotipo MIME del documento che sta spedendo. MIME sta per Multipurpose Internet Mail Extension ed è uno standard con cui si può specificare il tipo ed il sottotipo di un determinato documento. Il client, leggendo questa informazione, comprenderà che sta ricevendo un documento di tipo testo e in particolare un documento HTML, quindi andrà ad interpretare il documento scaricato come una pagina HTML, andando ad eseguire le formattazioni del caso.

Risposta HTTP (3/3)

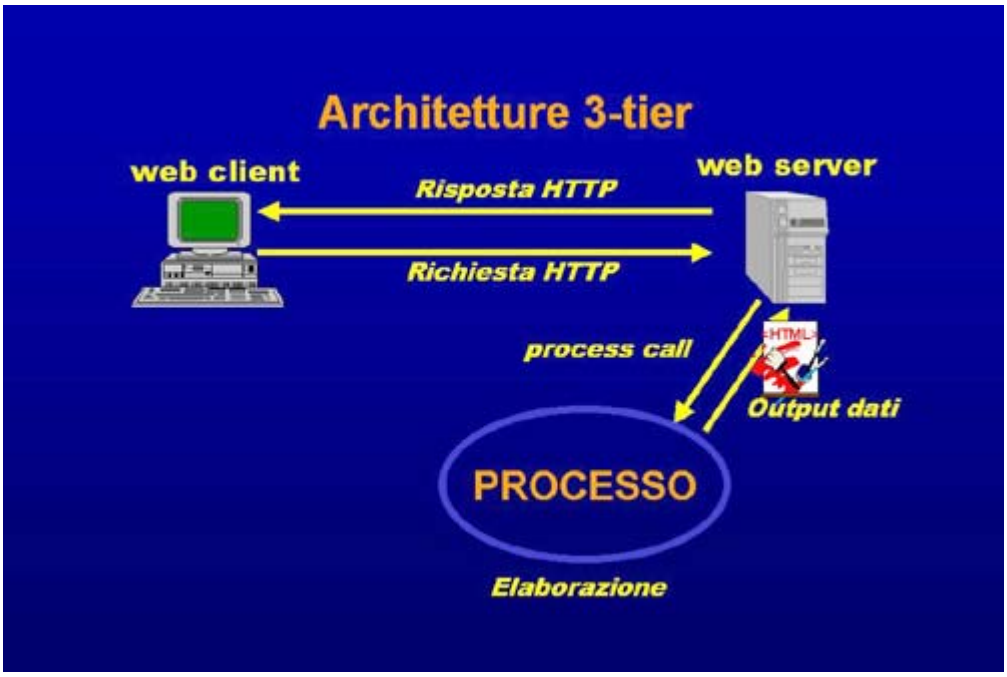
Ci sono, poi, tutta un'altra serie di informazioni che il server comunica al client: a cominciare dalla data in cui è avvenuta la connessione, in questo caso si tratta di una risposta che è stata spedita il 27 febbraio 2000 alle 10.00, alla data in cui è stato modificato per l'ultima volta questo documento. Questa informazione è particolarmente importante, in quanto consente al client di capire se eventuali copie locali di quel documento siano ancora aggiornate o siano state sostituite nel server stesso. Notate, infine, che il server specifica anche la dimensione del documento che sta trasferendo, in questo caso si tratta di un documento di 145 byte. In questo modo il client può sapere quale parte di documento ha già ricevuto.

Programmi CGI

Abbiamo visto in che modo il protocollo HTTP consente ad un client e ad un server di trasferire informazioni, di trasferire richieste e trasferire risposte. Il protocollo HTTP può essere utilizzato non solo per trasferire documenti HTML, ma una delle sue applicazioni più interessanti è la

possibilità di utilizzarlo per invocare programmi che risiedono sul server, a partire da un client Web. Questo aspetto di HTTP è particolarmente rilevante, in quanto consente di invocare applicazioni, attraverso un client Web, che si trovano su di un server Web. In questo caso sarà necessario specificare un protocollo con il quale il server Web possa attivare il processo richiesto, possa passare eventuali parametri necessari a quel processo e possa ricevere da quel processo l'output HTML da inviare al client. Questo protocollo è il protocollo CGI (Common Gateway Interface), che consente appunto ad un server Web di invocare un processo su richiesta di un client Web. Vedremo, in particolare, che questo meccanismo consente di sviluppare le cosiddette architetture a tre livelli. Architetture nelle quali l'accesso al sistema informativo avviene non più solo attraverso un client ed un server, ma avviene attraverso tre distinti livelli: il client Web, il server Web e il data base applicativo stesso. Vedremo che questa architettura ha una serie di vantaggi, che cercheremo di comprendere nell'ambito di questa lezione.

Architetture 2-tier



Che cosa significa architettura a tre livelli? Significa che possiamo utilizzare un normale client Web standard, senza bisogno di nessuna modifica, per andare ad effettuare una richiesta HTTP che richieda, non una semplice pagina HTML, bensì l'attivazione di un processo che risiede sul server Web. Questo processo potrà eseguire una propria elaborazione e potrà generare al volo i dati in output. Questi dati, ovviamente, dovranno essere formattati in HTML, in modo che il Web server possa girarli al client Web, che così otterrà una risposta alla propria interrogazione. Rispetto all'utilizzo di pagine HTML statiche, in questo caso abbiamo l'intervento di un processo che elabora al volo le informazioni, sulla base dei dati passati dal client HTTP. Abbiamo realizzato in questo modo, in tutto e per tutto, un meccanismo che consente di invocare delle procedure remote e che consente di eseguire dei processi su di un server Web. Dobbiamo comprendere in che modo un server Web possa attivare un processo CGI e come esso possa essere eseguito. In particolare, i server Web individuano nella richiesta HTTP la caratteristica che la richiesta sia relativa ad una pagina HTML, oppure ad un processo CGI. In particolare, se la richiesta si trova all'interno di una particolare directory, ad esempio la directory cgi-bin, il server la interpreterà come una richiesta di attivazione di un processo e non come la richiesta di distribuzione di una pagina. Questo processo verrà eseguito in locale e preparerà il proprio output per il client stesso.

http://130.192.3.92/cgi-bin/time.pl (1/2)

Andiamo a vedere come è possibile scrivere un programma CGI, per analizzarne quali sono le caratteristiche e quali sono le potenzialità. In particolare analizzeremo un semplice programma CGI scritto in linguaggio Perl. Perl è un linguaggio interpretato, molto utilizzato nelle applicazioni CGI, che consente di scrivere in maniera compatta e semplice programmi CGI. In questo caso stiamo andando a vedere il codice del programma che corrisponde all'indirizzo 130.192.3.92/cgi-bin/time.pl. Si tratta di una semplice applicazione CGI in cui il programma invocato va semplicemente a stampare la data e l'ora corrente, formattando in HTML per presentarla al client. Andiamo a vedere come è organizzato questo tipo di programma CGI. Allora, in questo caso, il programma deve specificare con quale tipo di interprete debba essere trattato il programma stesso e la prima riga di questo programma ha esattamente questa funzionalità. Dice al server Web qual è l'interprete da utilizzare per interpretare il programma descritto di seguito. In questo caso viene utilizzato il programma perl.exe, che si trova nella directory perl/bin.

http://130.192.3.92/cgi-bin/time.pl (2/2)

Notate che il programma CGI, a questo punto, ha il pieno controllo della connessione HTTP e deve occuparsi di tutti gli aspetti della connessione. In particolare, siccome il programma CGI è l'unico a conoscere il tipo di documenti che verrà generato da questa connessione, dovrà specificare che tipo di documento viene generato. In questo caso, vedete, questa istruzione di print specifica che il content-type del documento generato è di tipo text/html. Notate che la sintassi è la stessa che abbiamo visto qualche slide fa ed è la sintassi utilizzata e prevista dal protocollo HTTP per specificare il tipo di documento trasferito. A questo punto il CGI, vedete, stampa una coppia di ritorno a capo: questo fa si che venga passata dal server al client una riga contenente solo un carattere di ritorno a capo. Quindi, il client interpreta le righe successive come il body della risposta HTTP, che dovranno contenere delle informazioni HTML così come specificato dall'header content-type. Notate che, a questo punto, il programma CGI deve soltanto occuparsi di descrivere l'output HTML generato al volo a seconda dei dati corrispondenti a questo istante. In questo caso, il nostro programma CGI deve semplicemente stampare la data corrente. Allora, vedete, che andrà a scrivere semplicemente i tag HTML necessari a formattare la nostra pagina e poi invocherà la funzione localtime() che stamperà la data e l'ora corrente. Quindi, il programma CGI eseguirà un'altra print, per andare a stampare la chiusura della pagina HTML stessa.

http://130.192.3.92/cgi-bin/time.pl (test)

In questo modo abbiamo visto come funziona il protocollo CGI e come possiamo invocare una risorsa che si trova su server Web. Spostiamoci sul computer da questo lato e vediamo a quali fasi corrispondono questi procedimenti. Utilizziamo questo client Web e andiamo a collegarci ad un server Web, che in questo caso risiede sulla stessa macchina, ma questo è un dettaglio, potrebbe trovarsi in un qualsiasi punto della rete Internet. Andiamo ad accedere alla risorsa che abbiamo visto descritta nella slide precedente. In questo caso probabilmente non si legge, ma andiamo ad indicare che vogliamo accedere all'URL cgi-bin/time.pl, che individua in maniera univoca esattamente il file Perl che abbiamo visto poco fa. Invocando questa risorsa il programma CGI viene attivato dal server Web e vedete che, in questo caso, il programma stamperà un output in formato HTML che conterrà la stringa data e ora, così com'era indicato nel programma Perl che abbiamo visto e conterrà l'output del comando localtime(), che in questo caso riporta semplicemente l'ora in cui è stata eseguita questa connessione. In questo modo abbiamo visto com'è possibile attivare un processo che si trovi su di un server Web, andando ad invocarlo attraverso un client Web.

Metodi HTTP

Andiamo a vedere quali sono i meccanismi che HTTP ci mette a disposizione per attivare processi di questo tipo e, soprattutto, in che modo sia possibile passare dei parametri a questi processi. È indispensabile, infatti, quando eseguiamo elaborazioni in tempo reale, che l'utente non solo sia in grado di attivare un processo, ma sia anche in grado di inviare dei dati a quel processo. In particolare, HTTP fornisce due metodi per attivare processi: il primo va sotto il nome di metodo GET e il secondo va sotto il nome di metodo POST. Nell'utilizzo del metodo GET il server riceve eventuali parametri dal client attraverso il campo QUERY_STRING, che viene aggiunto, come abbiamo visto precedentemente, in coda alla URL e li passa al processo CGI attraverso una variabile d'ambiente che si chiama, appunto, QUERY_STRING. Nel caso di utilizzo del metodo POST, invece, il server riceve eventuali parametri direttamente nel body della richiesta HTTP che, se vi ricordate, non sempre è presente. Quando utilizzeremo il metodo POST e quando passeremo parametri dal client al server, la parte body sarà presente e conterrà, appunto, i parametri che il client passa al server. Il server, quindi, gira i parametri così ricevuti al CGI invocato, utilizzando lo standard input di quel processo. Un processo CGI invocato tramite metodo POST, andrà a leggere il proprio standard input per ritrovare i parametri che gli sono stati passati dal server Web.

Form HTML

Occorre specificare un modo con il quale il client possa raccogliere dei dati dall'utente e con il quale questi dati possano essere organizzati e passati al server Web. Siccome si tratta di organizzare il formato di dati, dovremmo utilizzare a questo scopo l'HTML, che in effetti prevede una serie di tag che consentono proprio di inserire dei dati in una pagina HTML e di passarli, attraverso la connessione HTTP, al server Web. La sintassi prevista dall'HTML utilizza la tag <FORM> che consente di specificare una serie di parametri da inviare al server. La tag <FORM> è caratterizzata da due parametri: il parametro ACTION, che specifica qual è l'URL obiettivo di questa azione, ovvero quale URL verrà invocato nell'eseguire questa form, e il metodo utilizzato. In questo caso, vedete, i parametri verranno passati dal client al server utilizzando il metodo GET. Nella form dobbiamo essere in grado di specificare quali sono i campi che devono essere passati al server e, in particolare, si utilizza una tag INPUT che ha tutta una serie di varianti, delle quali ne vediamo un paio, con la quale è possibile passare dei valori al server. Nel primo caso vediamo una tag INPUT di tipo TEXT, che consente di passare dei dati testuali dal client al server. Questi dati saranno identificati da un nome, così come specificato nel parametro NAME. La penultima riga di questa pagina HTML specifica, invece, un elemento di input di tipo BUTTON: un bottone la cui pressione provocherà l'esecuzione dell'azione specificata nel parametro ACTION della tag FORM.

http://130.192.3.92/formget.html

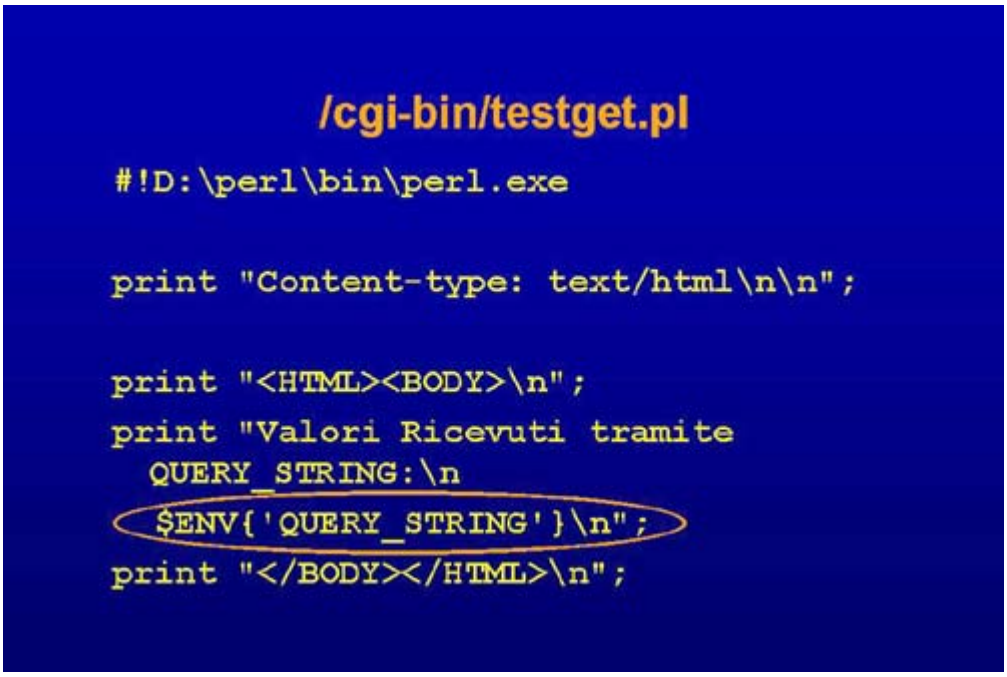
Questi sono tutti gli elementi che abbiamo a disposizione per organizzare la raccolta di dati sul client. Alla pressione del pulsante BUTTON questi verranno inviati sul server. Andiamo a vedere un esempio, semplice, in modo da comprendere esattamente quali sono i meccanismi utilizzati. In questo esempio andremo a vedere una semplice pagina HTML, la pagina formget.html, nella quale vedete specificata una form, la quale invoca il file cgi-bin/testget.pl, uno script Perl che utilizza il metodo GET. Notiamo che questa form prevede due parametri che vengono passati dal client al server: entrambi sono parametri di tipo testo, il primo si chiamerà var1 e il secondo si chiamerà var2. Infine, la form contiene un pulsante BUTTON che consente di inviare semplicemente i dati dal client al server.

http://130.192.3.92/formget.html (test)

Ci spostiamo sul PC e andiamo a vedere come funzionano questi esempi. Cominciamo dal scaricare la pagina HTML che descrive la form che intendiamo utilizzare. In questo caso vado a scrivere

formget.html per individuare la pagina HTML descritta dalla form che abbiamo appena visto. Come ci aspettavamo, questa form ci presenta due campi: sono due campi di tipo testo in cui possiamo inserire due valori. In questo caso inserisco nel campo var1 il valore 1 e nel campo var2 il valore 2. Notate che quando, premendo il pulsante che si trova in fondo a questa form, invio questi dati, verrà invocato il processo CGI indicato dal parametro ACTION della form, il quale elaborerà l'input ricevuto. Non si può leggere perché i caratteri sono molto piccoli, ma nella location bar del mio navigatore sono stati aggiunti esattamente il valore 1 e il valore 2 che ho inserito nella form precedente. Questo ha consentito al client di inviare, attraverso il metodo GET, i parametri al server. E notate che, in effetti, il processo CGI invocato ha ricevuto dei valori tramite la variabile d'ambiente QUERY_STRING e i parametri ricevuti sono esattamente quelli che avevo inviato: valore 1 e valore 2. I dati, in questo caso, sono raccolti in coppie del tipo nome = valore, dove il nome è quello specificato nell'attributo NAME della tag input.

/cgi-bin/testget.pl



Il programma CGI che eseguirà questo tipo di elaborazione è quello riportato in questa slide. Vedete che il programma CGI, dopo aver specificato che andrà a produrre un output di tipo text/html, stamperà semplicemente il valore della variabile di environment QUERY_STRING. Quella indicata qui è la sintassi con la quale il Perl consente di stampare il valore di una variabile di environment, il cui nome è indicato in questa stringa. Questo è esattamente il valore dei due parametri che abbiamo passato al nostro server Web.

http://130.192.3.92/formpost.html

Procediamo in questa analisi del protocollo HTTP e del protocollo CGI, cercando di comprendere qual è l'altro meccanismo che consente di passare parametri da un client ad un server. Abbiamo appena visto come funziona il metodo GET e abbiamo visto che il programma CGI accede ai parametri passati dal client attraverso una variabile di ambiente, che ha il nome di QUERY_STRING. Vediamo come funziona il metodo POST. Osserviamo come è organizzata la pagina HTML che consente di preparare la raccolta dei dati. In questo caso, come vedete, il metodo utilizzato è il metodo POST e verrà invocato uno script CGI che va sotto il nome di testpost.pl. La form passa poi esattamente gli stessi parametri che abbiamo visto prima: ovvero due parametri di tipo text dal nome var1 e var2.

http://130.192.3.92/formpost.html (test)

Ci spostiamo sul PC per vedere come questa pagina HTML possa essere usata per invocare lo script CGI, utilizzando questa volta il metodo POST. Andiamo a caricare la pagina HTML

corrispondente: indicherò questa volta che l'URL al quale sono interessato è formpost.html. Eseguo il reload di questa pagina in modo da avere la nuova pagina a disposizione e vado ad inserire i due valori. In questo caso passo i valori val1 e val2 nei due campi var1 e var2. Invio questa form, che questa volta utilizzerà il metodo POST come meccanismo di trasporto dei parametri, i quali saranno passati allo script CGI attraverso lo standard input. Se guardate con attenzione nella location bar, questa volta vediamo che non è presente nessuna QUERY_STRING in coda all'URL, proprio perché il meccanismo di trasporto utilizzato in questo caso è quello del metodo POST. Questi parametri, i valori val1 e val2, sono stati passati nel body della connessione HTTP e non nel campo QUERY_STRING dell'URL.

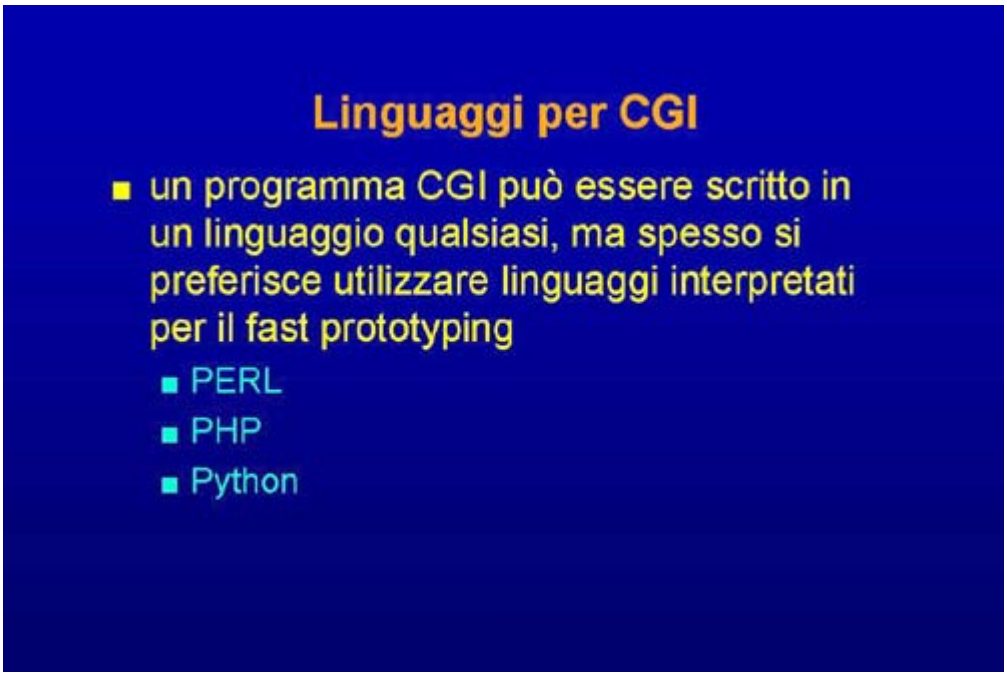
/cgi-bin/testpost.pl

Possiamo spostarci di nuovo sulla slide e andremo a vedere qual è in questo caso il codice del programma CGI che gestisce questa richiesta. In questo caso, vediamo ancora un semplice programma Perl in cui i parametri, questa volta, non vengono letti attraverso una variabile d'ambiente, ma vengono letti andando a leggere lo standard input. Questa è la riga che in Perl consente di leggere una riga dallo standard input. In questo caso, la variabile $param conterrà esattamente i parametri che il client ha passato al server Web utilizzando il metodo POST.

URL encoding

Esiste un problema che non abbiamo considerato, ma che va menzionato: il protocollo HTTP utilizza dei caratteri con significato particolare. Dobbiamo, quindi, prevedere un meccanismo per cui se l'utente inserisce uno di questi caratteri particolari, questi vengano transcodificati: ovvero vengano convertiti in caratteri trasmissibili senza problemi sulla connessine HTTP. Utilizziamo l'esempio che abbiamo visto prima e passiamo come valore al nostro script CGI una variabile che contenga uno spazio e un carattere ;, che sono entrambi caratteri speciali. Ci accorgiamo che inviando questi valori dal client al server, il client effettuerà una transcodifica, ovvero andrà a convertire questi caratteri particolari, lo spazio e il punto e virgola, in due valori che siamo trasmissibili sul protocollo HTTP. Questo procedimento va sotto il nome di URL encoding ed è essenzialmente una procedura eseguita dal client prima di trasmettere caratteri particolari. Come riportato in questa slide, viene eseguita transcodifica per gli spazi (che vengono transcodificati con il carattere +), per i segni di interpunzione (esclusi i caratteri indicati qui tra parentesi) e per i caratteri ASCII, che vengono sostituiti con il loro codice ASCII esadecimale preceduto dal carattere %. Nell'esempio prima avevamo visto che il carattere ; era stato sostituito dalla sequenza %3b. Questo significa che il codice ASCII esadecimale del carattere ; è il valore 3b. Il programma CGI dovrà quindi occuparsi, non soltanto di ricevere i parametri con il metodo GET o con il metodo POST, ma dovrà occuparsi anche della loro decodifica. Dovrà andare a riconvertire correttamente quei caratteri che hanno subito URL encoding.

Linguaggi per CGI



Questa analisi che abbiamo portato al termine, ci ha consentito di comprendere quali sono i meccanismi che regolano il protocollo HTTP e in che modo il protocollo HTTP possa essere utilizzato per invocare processi remoti. In particolare, abbiamo visto quali sono i metodi utilizzati dal protocollo HTTP per invocare processi remoti e abbiamo visto alcuni esempi CGI scritti in linguaggio Perl. Perl non è l'unico linguaggio utilizzabile per il programmi CGI, ma anzi può essere utilizzato un linguaggio qualsiasi. Possono anche essere utilizzati i linguaggi compilati, come ad esempio il C o il C++, ma generalmente si preferisce utilizzare linguaggi interpretati, perché questo consente di sviluppare più rapidamente prototipi. Tra il linguaggi più diffusi vi è sicuramente il Perl. Altri linguaggi molto utilizzati per la programmazione CGI sono il linguaggio PHP ed il linguaggio Python, che sono delle varianti di Perl che si avvicinano al linguaggio C. Quello che abbiamo visto in questa lezione è quindi un introduzione al protocollo HTTP. Nelle prossime vedremo quali sono le altre architetture a tre livelli e quali sono le caratteristiche di ciascuna di essa.