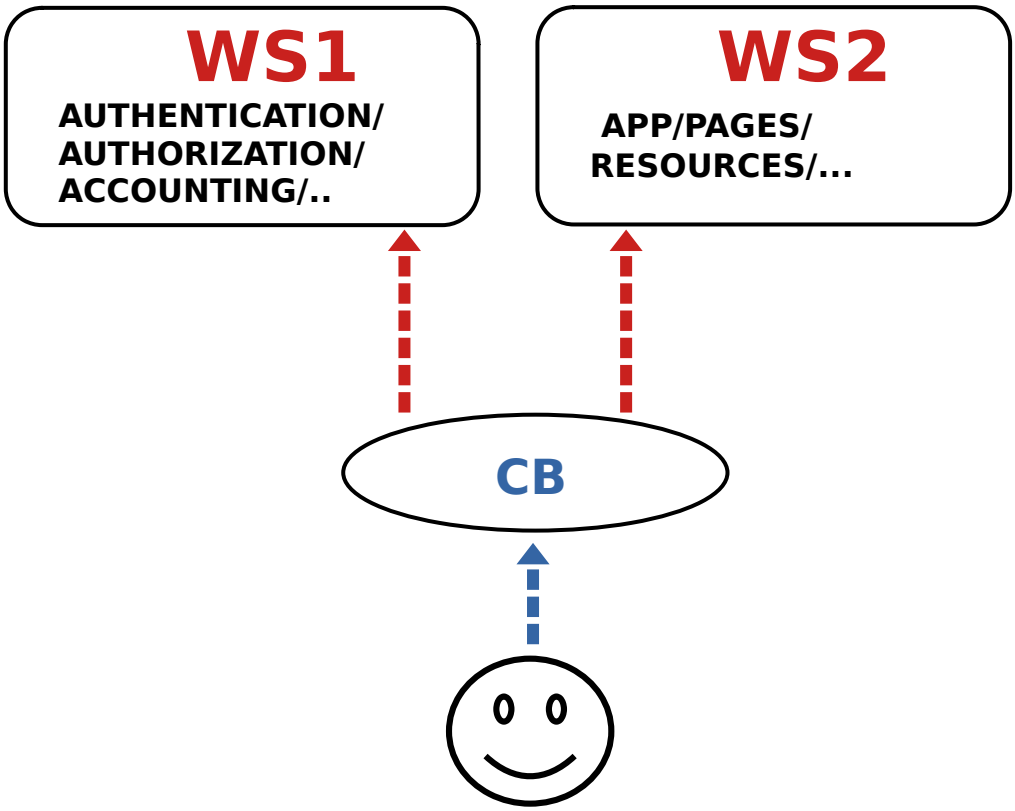
Progettazione di un’Applicazione Web

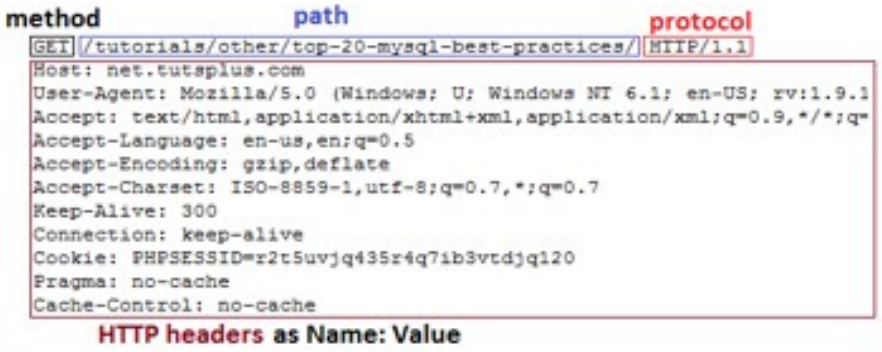
Consideriamo di progettare un’Applicazione Web il cui funzionamento è garantito da due Web Server.
Il primo Web Server(WS1) detto Server di Autenticazione si occupa di svolgere determinati servizi che verranno descritti in seguito.
Il secondo Web Server(WS2) invece si occupa invece di garantire le funzionalità dell’applicazione come ad esempio fornire contenuti di risposta alle richieste dei Client(che potrebbero essere ad esempio delle pagine Web interattive).
Nella progettazione di WS1 elabora dei meccanismi per realizzare le funzioni “AAA”:

1)Autenticazione: processo con cui identificare il Client
2)Autorizzazione: concessioni ai client di privilegi di vario genere(tempo di connessione,larghezza di banda riservata,risorse accessibili...)
3)Accounting: monitoraggio del consumo di risorse da parte dei client. Quest’ultima attività ha vari scopi che vanno oltre il semplice controllo,come l’ottimizzazione)



Supponiamo che i vari utenti accedano a questi servizi attraverso un Client Browser(ad es. google chrome).
Fondamentalmente il Browser permette che il Client possa comunicare con il Server e si pone in un certo senso come intermediario tra i due facendo varie operazioni.
Tra queste operazioni una delle più importanti è sicuramente quella di effettuare Richieste HTTP che sono i messaggi con cui si comunica nel Livello Applicativo.Il browser si occupa di elaborare le informazioni richieste dal client e le inserisce in maniera opportuna all’interno di una richiesta HTTP.

Richiesta HTTP



Osserviamo che nell'header di una richiesta HTTP ci sono una serie di campi interessanti utili a comprendere il funzionamento di un Web Server e a capire la differenza con il modello Client-Server originale.

Il metodo(GET,POST,PUT,OPTION...) specifica il tipo di richiesta che il Client vuole fare.
Il path(URL) specifica la risorsa a cui il Client vuole accedere(ad.esempio una pagina web o i record di una base di dati).
Insieme all'URL viene specificato la versione del protocollo HTTP con cui viene fatta la richiesta.
Un altro campo importante è l'host a cui si vuole effettuare la richiesta.Le recenti tecnologie di VirtualHosting permettono di registrare più host allo stesso IP.Dunque l'URL della risorsa unito al nome dell'host fanno sì costituiscono un URI.
Infine nel campo cookie si osserva una stringa detta Session-Id,un identificativo con cui l'utente accede alla propria sessione.Ricordiamo che il Web server non memorizza la sessione di ogni utente a differenza dei Server nel modello originale.

AUTENTICAZIONE

Supponiamo che il Client si autentichi inserendo dei campi(username e password) in una pagina.
Sarà il browser che invierà al Server una richiesta HTPP inserendo nell'URL Username e Password che ha digitato il Client.

Cosa succede se il CB(cioè il client attraverso il Browser) effettua una richiesta al WS2 senza essersi autenticato attraverso il WS1?
WS2 risponde con un messaggio HTTP con codice 302 FOUND e con PATH che redirige alla pagina di autenticazione con le opportune queries(http:/....? QUERY).Nella query viene salvata la risorsa chiesta inizialmente dal CB.
Il CB in automatico risponde effettuando la richiesta questa pagina.
Osserviamo che WS2 e WS1 non hanno comunicato in maniera diretta.
WS2 ha inviato una risposta a CB che lo ha fatto collegare a WS1.
Ora il WS1(detto anche IDENTITY MANAGER) che ha ricevuto la richiesta dal CB gli fa effettuare il login.Avendo salvato la risorsa che CB aveva richiesto inizialmente riformula una richiesta HTTP inserendo nella query un identificativo per il client(token e id).Il CB inoltrerà questa richiesta a WS2.
Solo in questo passo il WS2 e ID(WS1) comunicano direttamente perché il primo chiede al secondo se token e id siano corretti.
Questi ultimi due passaggi si possono fare diversamente utilizzando una chiave per l'autenticazione a WS2 piuttosto che una coppia token-id da verificare.

Come è fatto il token?(JSON WEB TOKEN)
Ricordiamo che il protocollo HTTP è stateless(cioè il Server è privo di memoria).

Cosa sono i cookie?
Sono oggetti del tipo chiave-valore.Possono essere memorizzati all'interno di files o databases perché il Browser deve poterli riutilizzare.I cookies sono gestiti a livello applicativo dai Browser.Nel cookie ci sono tutte informazioni utili al Browser e al Web Server.
Se WS1 mi ha dato 10 cookies,li presento solo quando mi connetto ad esso.
Il Web Server dà al CB vari cookies per varie utilità:

1) un cookie che gestisce le autorizzazioni.
2) un cookie che gestisce lo username.

Come viene gestista la sicurezza del cookie?

3) un cookie che contiene la data di quando è stato dato il cookie.Facendo una set-cookie entro tot tempo posso prolungarne la durata.

Quando un CB si collega a un Web Server gli consegna tutti i cookies che quest'ultimo gli ha dato.

Perché nel Web moderno è difficile effettuare log-out?
Innanzitutto il modello HTTP è stateless e secondo poi nel log-out il WS dovrebbe eliminare tutti i cookie creati per quel CB.Il WS non ricorda né dei login e dei logout del CB altrimenti dovrebbe tenere troppa memoria.Ma la peculiarità dei WS è non mantenere memoria per i client.
Molti implementano il logout facendo la setCookie a vuoto(ma non sempre funziona).
Se ci si collega con lo stesso user da due Browser differenti,il Web Server se ne accorge perché ha dato a questi due cookies differenti.