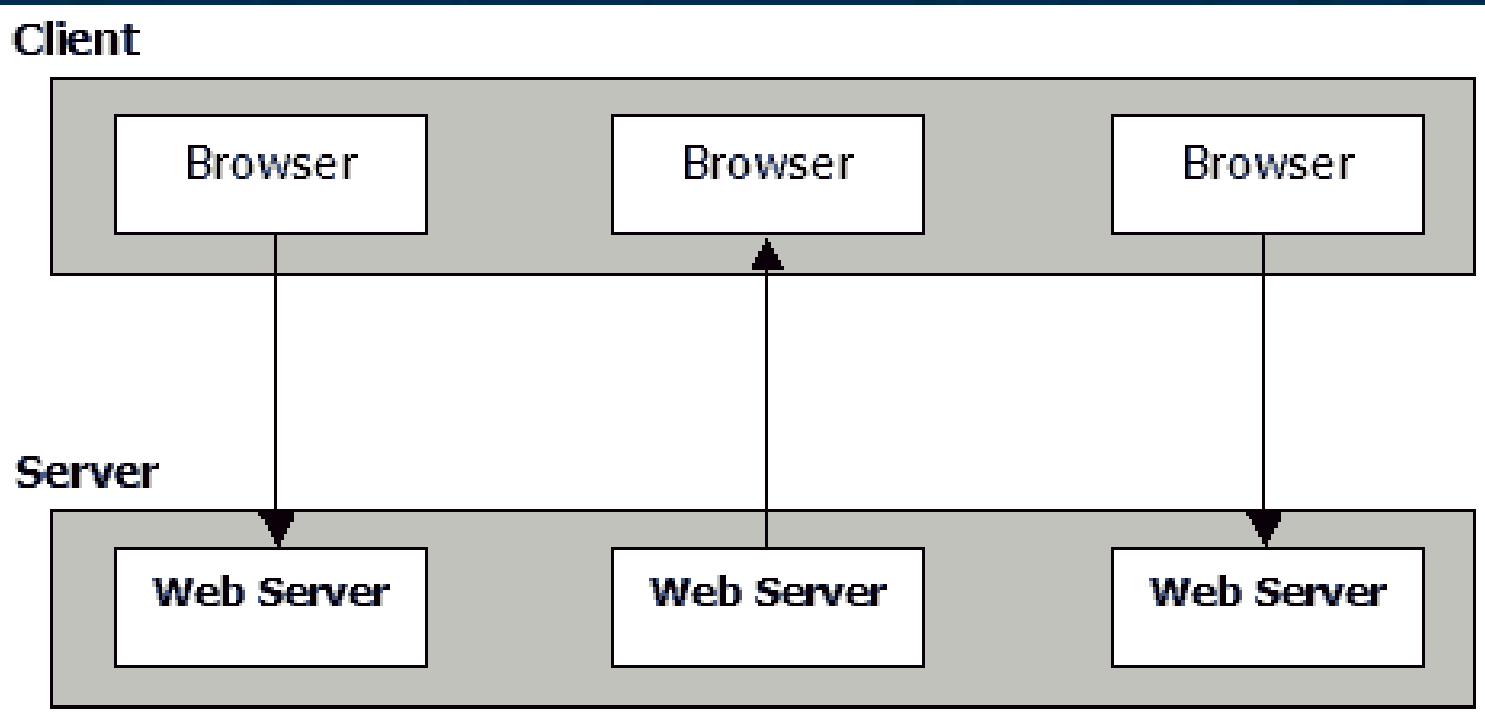
Session tracking

Session tracking

- HTTP: è stateless, cioè non permette di associare una sequenza di richieste ad un dato utente.
- Ciò vuol dire che, in generale, se un browser richiede una specifica pagina web situata su un determinato web server, quest'ultimo, dopo aver soddisfatto la richiesta, chiude la connessione con il client e non conserva alcuna informazione sul client stesso. Pertanto, se lo stesso browser dovesse effettuare successive richieste allo stesso web server, questo non avrebbe alcun modo per associare il browser con le richieste pervenute in precedenza.



Prima richiesta HTTP:

Il Browser effettua la richiesta di una pagina

Prima risposta HTTP:

Il server restituisce la pagina richiesta e chiude la connessione con il client

Successive richieste HTTP:

Il browser effettua la richiesta di una pagina. Il server non ha alcun modo per associare il browser con le richieste precedenti pervenute dallo stesso client.

Session tracking

Cos'è una sessione:

- Tenere traccia degli utenti durante la navigazione su un determinato sito rappresenta quello che, tecnicamente, viene definito "Sessione Web".

Session tracking

- Esempio: si pensi all'applicazione e-commerce. E' necessario memorizzare lo stato per "ricordare" l'history dell'utente !!!!
- Tenere traccia di un utente su un sito di e-commerce vorrà dire, tra le altre cose, non perdere le informazioni relative agli articoli già scelti per l'acquisto durante la navigazione (e consentire all'utente stesso di poter scegliere ulteriori articoli) tra una pagina e l'altra.

Session tracking methods

- User authorization
- Cookies
- URL rewriting
- Hidden fields
- Session tracking API
- SSL

User Authorization

Autenticazione utente:

processo che porta a riconoscere un client tramite user name e una password.

E’ possibile dare l’accesso ad alcune risorse solo a utenti identificati da user name e password noti.

User Authorization

- Vantaggi
  - Facile da realizzare
  - Indipendente dalla macchina dalla quale si connette
- Svantaggi
  - Registrazione obbligatoria
  - Login obbligatorio
  - Logout solo disconnettendosi dal browser

Cookies

Cookie:

è un’informazione inviata al browser da un server web. Quando il client riceve un cookie lo salva e poi lo rinvia al server ogni volta che accede a una pagina su di esso.

Cookies

Un cookie ha un nome, un valore, e alcuni attributi facoltativi come una descrizione, una data di scadenza, un numero di versione e così via.

Cookies

I cookie vengono ritrasmessi verso il server HTTP ogni volta che:

- il cookie non è scaduto
- il dominio che interroghiamo è il dominio di persistenza del cookie

Cookies

Limitazioni: i browser accettano

- 20 cookie per sito
- 300 cookie per utente
- 4096 byte per cookie

Cookies

- Vantaggi
  - Anonimo e trasparente
- Svantaggi
  - Fortemente legato alla macchina, i cookie sono memorizzati su disco !!!
  - Il browser può non supportare i cookie

URL Rewriting

I dati che identificano la sessione vengono appesi alla fine dell'URL.

URL originale:

http://server:port/servlet/servletName

URL riscritto:

http://server:port/servlet/servletName?sessionid=7456

URL Rewriting

- Vantaggi
  - Funziona anche se i cookies vengono disabilitati
  - Tutti i dati sono appesi all'url =>facile fare il debug.
- Svantaggi
  - La lunghezza dell'url è limitata a 2000 caratteri
  - Assenza di riservatezza dei dati
  - La sessione viene persa se si naviga verso sezioni statiche dello stesso sito

Hidden Fields

I form HTML hanno un tag di questo tipo:

<INPUT TYPE>=”HIDDEN” NAME=”SESSION” VALUE=”...”

- Quando il form viene sottomesso il nome ed il valore specificati sono inclusi nei dati del methodo GET o POST
- Questi campi non sono visibili direttamente dall'utente

```
<html>
<head>
<title>My Page</title>
</head>
<body>
<form name="myform"
   action="http://www.mydomain.com/myformhandler.cgi"
   method="POST">
<div align="center">
<input type="text" size="25" value="Enter your name here!">
<input type="hidden" name="Language" value="English">
<br><br>
</div>
</form>
</body>
</html
```

Hidden Fields

- Vantaggi
  - Funziona anche se i cookies vengono disabilitati
  - E' possibile utilizzare il metodo HTTP POST, bypassando, così, il problema dello spazio limitato a 2000 caratteri.
- Svantaggi
  - Funziona solo se ogni pagina e generata dinamicamente

Session tracking Api

Che cos'è una Servlet?

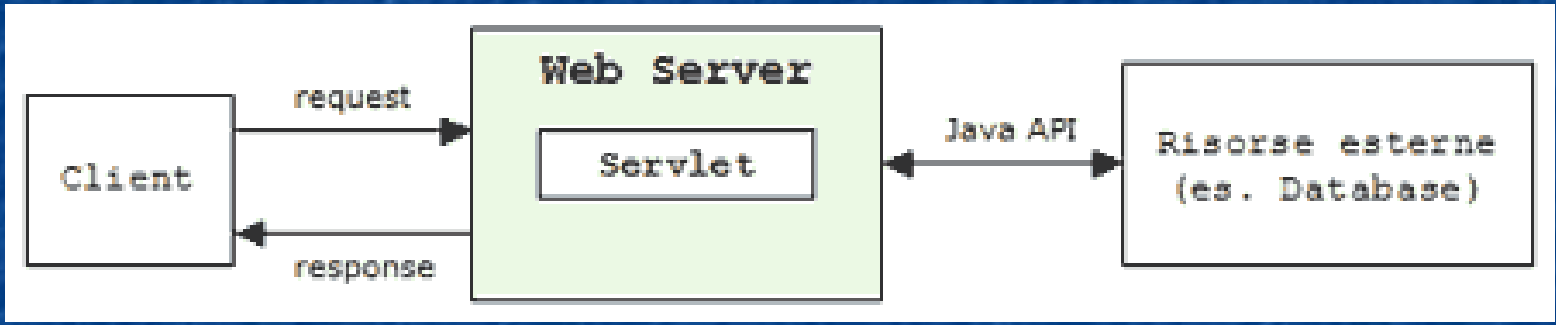
- Semplicemente, un programma scritto in Java e residente su un server, in grado di gestire le richieste generate da uno o più client, attraverso uno scambio di messaggi tra il server ed i client stessi che hanno effettuato la richiesta. Tipicamente sono collocate all'interno di Application Server o Web Application Server come, ad esempio, Tomcat.

Session tracking Api

- Visto che le Servlet sono scritte in Java esse possono avvalersi interamente delle Java API che, come è noto, consentono di implementare con relativa semplicità anche svariate funzionalità complesse e importanti come, ad esempio, l'accesso ad un database.
- Ad esse si aggiungono, poi, le più specifiche servlet API, che mettono a disposizione un'interfaccia standard utilizzata per gestire la comunicazione tra un client Web ed una Servlet.

Session tracking Api

Il flusso di una Servlet:

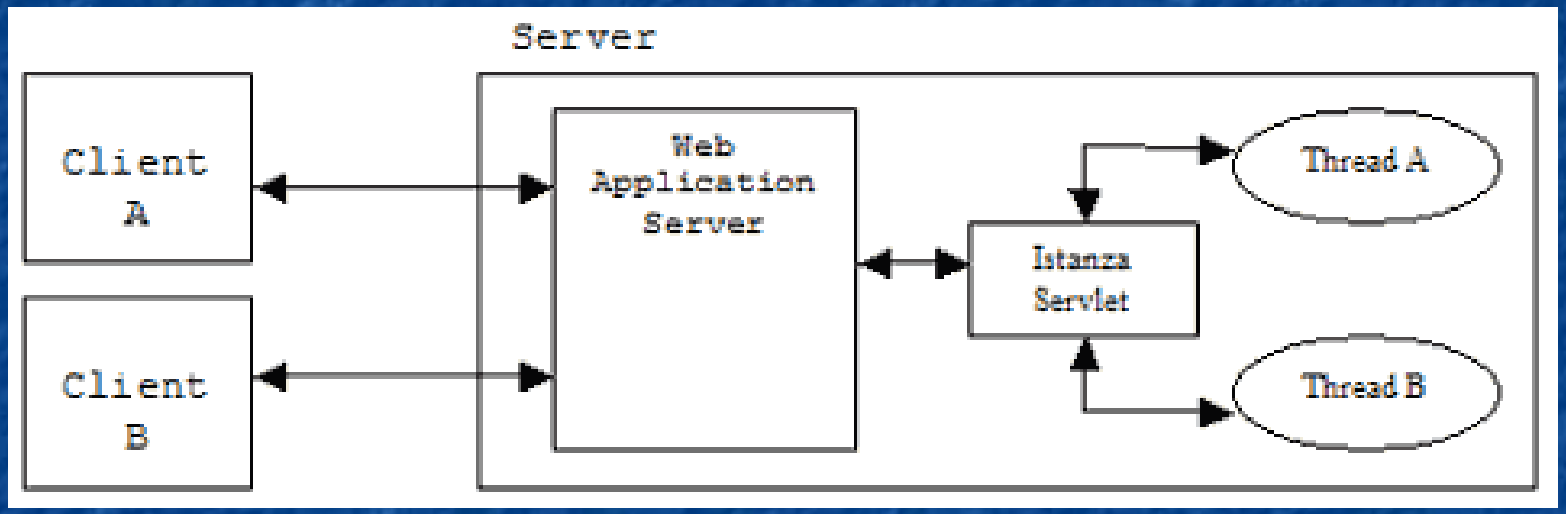


- Un client invia una richiesta (request) per una servlet ad un web application server.
- Qualora si tratti della prima richiesta, il server istanzia e carica la servlet in questione avviando un thread che gestisca la comunicazione con la servlet stessa.

Session tracking Api

- Nel caso, invece, in cui la Servlet sia già stata caricata in precedenza (il che, normalmente, presuppone che un altro client abbia effettuato una richiesta antecedente quella attuale) allora verrà, più semplicemente, creato un ulteriore thread che sarà associato al nuovo client, senza la necessità di ricaricare ancora la Servlet.
- Il server invia alla servlet la richiesta pervenutagli dal client.
- La servlet costruisce ed imposta la risposta (response) e la inoltra al server.
- Il server invia la risposta al client.

Session tracking Api



Session tracking Api

Abbiamo visto, esaminando il flusso di esecuzione di una servlet, che il flusso stesso è incentrato su due componenti fondamentali:

- la richiesta (request, inviata dal client verso il server)
- la risposta (response, inviata dal server verso il client).

In Java, questi due componenti sono identificati, rispettivamente, dalle seguenti interfacce:

- javax.servlet.http.HttpServletRequest
- javax.servlet.http.HttpServletResponse

Session tracking Api

- Un oggetto di tipo HttpServletRequest consente ad una Servlet di ricavare svariate informazioni sul sistema e sull'ambiente relativo al client.
- L'oggetto di tipo HttpServletResponse, invece, costituisce la risposta da inviare al client e che, come si è detto, può essere dipendente da svariati data source.

Session tracking Api

Creare una Servlet vuol dire, in termini pratici:

- Definire una classe che derivi dalla classe HttpServlet. I metodi più comuni per molti dei quali si è soliti eseguire l'overriding nella classe derivata sono i seguenti:
- void doGet(HttpServletRequest req, HttpServletResponse resp)
  Gestisce le richieste HTTP di tipo GET. Viene invocato da service()
- void doPost(HttpServletRequest req, HttpServletResponse resp)
  Gestisce le richieste HTTP di tipo POST. Viene invocato da service()

Session tracking Api

Qualcuno potrebbe domandarsi per quale motivo possa essere più vantaggioso utilizzare le Servlet piuttosto che affidarsi a tecnologie, ancora abbastanza utilizzate, come la Common Gateway Interface (CGI). Beh, le differenze non sono trascurabili. Tra le più evidenti possiamo citare:

Portabilità: Grazie alla tecnologia Java, le servlet possono essere facilmente programmate e "portate" da una piattaforma ad un'altra senza particolari problemi.

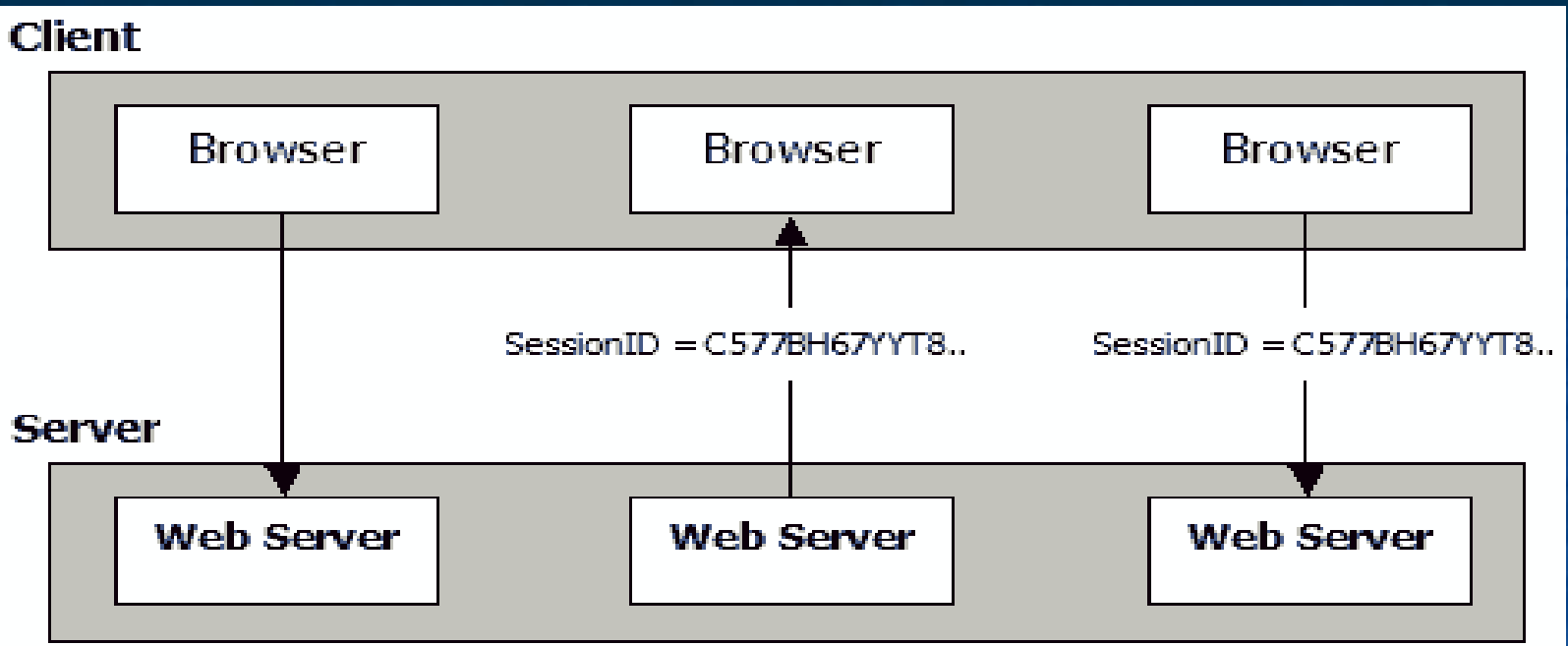
Session tracking Api

Efficienza. Come abbiamo detto le servlet vengono istanziate e caricate una volta soltanto, alla prima invocazione. Tutte le successive chiamate da parte di nuovi client vengono gestite creando dei nuovi thread che si prendono carico del processo di comunicazione (un thread per ogni client) fino al termine delle rispettive sessioni. Con le CGI, tutto questo non accadeva. Ogni client che effettuava una richiesta al server causava il caricamento completo del processo CGI con un degrado delle performance facilmente immaginabile.

Session tracking Api

Persistenza: Dopo il caricamento, una servlet rimane in memoria mantenendo intatte determinate informazioni (come la connessione ad un data base) anche alle successive richieste.

Gestione delle sessioni: Come è noto il protocollo HTTP è un protocollo stateless (senza stati) e, pertanto non in grado di ricordare i dettagli delle precedenti richieste provenienti da uno stesso client. Le servlet sono in grado di superare questa limitazione.



Prima richiesta HTTP:

Il browser richiede una pagina JSP o una Servlet. Il Servlet Engine (Tomcat, ad esempio) crea un oggetto sessione e assegna un ID univoco alla sessione stessa

Prima risposta HTTP:

Il server restituisce al browser la pagina richiesta insieme ad un identificativo per la sessione (ID).

Successive richieste HTTP:

Il browser richiede una pagina JSP o una Servlet. Il Servlet engine usa l'ID di sessione per associare il browser con l'oggetto sessione già presente sul server.

Session tracking Api

- Il browser richiede una pagina JSP o una Servlet inoltrando la richiesta ad un web server.
- Quest'ultimo non fa altro che passare la richiesta stessa al servlet engine, il quale controlla se la request pervenuta includa o meno un ID per la sessione. Se così non è, il Servlet Engine crea un identificativo univoco, istanziando anche un "session object" ad esso correlato che possa essere utilizzato per salvare i dati utili alla sessione.

Session tracking Api

- Da questo momento in poi il server utilizzerà l'ID di sessione (session ID) per mettere in relazione ogni richiesta del browser con l'opportuno oggetto di sessione precedentemente creato.
- Si noti che tutto questo avviene senza minimamente intaccare la logica stateless del protocollo HTTP, il quale continuerà a considerare chiusa la connessione con il client dopo aver soddisfatto ogni richiesta pervenuta.

Session tracking Api

- Per default viene utilizzato un cookie per il salvataggio del session ID sul browser del client. In tal modo ad ogni successiva richiesta del browser verrà accodato alla request, l'ID prelevato dal cookie.
- Le servlet API definiscono la classe **javax.servlet.http.Cookie** per la gestione dei cookie
- Una servlet recupera *tutti* i cookie che ha inviato attraverso il metodo

```
public Cookie[]
  HttpServletRequest.getCookies()
```

Session tracking

- Per default i cookie sono restituiti solo all'host che li ha salvati
- È possibile impostare la scadenza per un cookie utilizzando il metodo

public void Cookie.setMaxAge(int expiry)

Session tracking

- Vantaggi
  - Anonimo e trasparente
- Svantaggi
  - Fortemente legato alla macchina, i cookie sono memorizzati su disco !!!
  - Il browser può non supportare i cookie

Session tracking Api

- La soluzione alternativa più efficace è nota come URL encoding e fornisce un modo per riscrivere le URL in modo che esse contengano al loro interno anche il session ID. Grazie all'URL encoding è possibile tenere traccia delle sessioni anche nei casi in cui un utente abbia deciso di disabilitare i cookie sul proprio browser.

Session tracking Api

- È facile, in Java, ottenere un riferimento ad un session object.
- Nelle Servlet sarà sufficiente utilizzare il metodo getSession() dell'oggetto request per ottenere un riferimento ad un oggetto di tipo HttpSession:

```
HttpSession session = request.getSession();
```

Session tracking

- Attraverso il metodo getSession() verrà restituito l'oggetto di tipo HttpSession relativo alla sessione corrente, qualora ne esista già una. Nel caso in cui non fosse presente alcuna sessione, tale metodo si occuperà di crearne una nuova.

```
public void doPost(HttpServletRequest
req,HttpServletResponse res) {
....
    HttpSession session =
req.getSession(true);
}
```

Session tracking

I tre metodi classici utilizzati per lavorare con eventuali attributi del session object sono i seguenti:

public voi setAttribute(String name, Object value)

public Object getAttribute(String name)

public void removeAttribute(String name)

Session tracking

public voi setAttribute(String name, Object value)

consente di impostare un oggetto Java come attributo di una sessione, assegnandogli un nome. Ad esempio:

```
session.setAttribute("codiceArticolo", itemCode);
```

Session tracking

public Object getAttribute(String name)

permette di ricavare il valore di un attributo fornendo in input il nome dell'attributo stesso. Poiché il tipo restituito è di classe Object sarà necessario effettuare un casting per convertire il risultato al tipo desiderato

```
String itemCode = (String)
  session.getAttribute("codiceArticolo")
```

Session tracking

public void removeAttribute(String name)

removeAttribute() elimina un attributo dal session object, fornendo in input al metodo stesso il nome dell'attributo da rimuovere. Ad esempio:

session.removeAttribute("codiceArticolo")

Session tracking

Il metodo di URL Encoding:

- spesso tale metodo è anche denominato come "URL Rewriting", in quanto consente, attraverso l'invocazione di un particolare metodo, la riscrittura della URL di destinazione.
- L'URL Rewriting, al contrario dell'URL Encoding visto in precedenza, non utilizza le API messe a disposizione da Java per gestire la propria funzionalità.

Session tracking

- Lo scopo dell'URL Encoding è quello di aggiungere il session ID ad ogni URL facente capo ad un determinato sito, ogni qualvolta questa venga richiesta attraverso il browser. Per far ciò viene utilizzato il metodo encodeURL() dell'oggetto implicito response, così definito:

public String encodeURL (String url)

Session tracking

- Per impostare la action di una pagina JSP in modo che invochi una determinata servlet, ad esempio, si dovrà utilizzare la seguente sintassi:

```
<%
  String url =
   response.encodeURL("../servlet/MyServlet");
%>
<form action="<%=url%>" method="post">
```

Session tracking Api

- È importante ricordarsi di utilizzare il metodo urlEncode() su tutte le URL utilizzate all'interno dell'applicazione web che vengano invocate direttamente da una richiesta del browser.
- Se ci si scordasse di inserirne qualcuna lo scotto da pagare sarebbe quello di perdere traccia della sessione, con tutti i rischi che ne conseguirebbero.

SSL

- SSL: Secure Sockets Layer.
- Si interpone tra l'HTTP e il TCP/IP
- Si occupa della sicurezza con il ricorso della crittografia a chiave pubblica per lo scambio di chiavi simmetriche che cifrano la comunicazione client-server.
- La tecnologia SSL è stata sviluppata da Netscape

SSL

Funzionamento:

- L’utente si connette al sito sicuro SSL. I siti protetti con SSL hanno URL del tipo https://.... (HTTP più SSL)
- Il server firma la sua chiave pubblica con la sua chiave privata
- Il browser usa la chiave pubblica del server per verificare che la persona che ha firmato la chiave sia effettivamente il suo proprietario

SSL

- Il browser verifica se un'autorità di certificazione fidata ha firmato la chiave.

  Se non c'è questo tipo di firma il browser chiede all'utente di accettare o meno la firma.
- Il client genera una chiave simmetrica che viene cifrata con la chiave pubblica del server e rinviata a quest'ultimo.