Programmazione per il Web
Note per la lezione del 14/03/2016

Igor Melatti

Cookie, URL rewriting e Session Tracking

- Slide 151: il problema, in sintesi, sempre quello di introdurre lo stato sul protocollo HTML, che è stateless.
- Slide 152: non è una cosa che si può fare sul server. Si supponga di memorizzare, per ogni richiesta in ingresso proveniente da un certo host, le informazioni che si vogliono mantenere, ad esempio nei campi (ovviamente dei vettori...) di una servlet. Questo andrebbe bene se ci fosse un matching 1-1 tra utenti e host. Così non è, ad esempio se l'host è una workstation condivisa da più utenti contemporaneamente. O peggio ancora, se più utenti si "nascondono" dietro lo stesso proxy... Insomma, non ci si può fare affidamento.
  - Quindi si memorizzano (parte del)le informazioni *sempre* sul client browser. In questo modo, l'unica cosa che può andare storta è quando un utente lascia il proprio computer sbloccato ed incustodito con il browser già aperto...
- Slide 153: leggere anche il libro di testo a pagina 200, spiega che l'idea è quella di memorizzare sul cookie l'ID che si usa come chiave per memorizzare in una hashtable (o in una Map) tutte le informazioni su una data sessione. In questo modo, sul client browser c'è un ID che per il client non ha nessun senso (lo ha solo sul server), e tutte le info sono poi memorizzate in modo strutturato dentro un'hashtable sul server
  - occorre usare un'hashtable per ogni utente, per memorizzare coppie (chiave-valore), com'era per i cookies
  - siccome bisogna anche memorizzare l'accoppiata (id utente, hashtable), occorre un'hashtable di hashtables
  - come spiegato poco più avanti dal libro stesso, questo modo di progettare è talmente comune che viene direttamente realizzato con le sessioni: il programmatore deve solo pensare all'oggetto che rappresenta le scelte dell'utente corrente

  - contrariamente a quello che potrebbe sembrare leggendo il libro, anche PHP e ASP gestiscono le sessioni, e lo fanno allo stesso modo...

- Differentemente dai cookie, non si può settare il dominio, e quindi le sessioni funzionano solo all'interno della stessa context root
- Esempio `slide159`: come si possono sostituire ai cookie le sessioni
- Slide 188: si può invocare un qualsiasi URL appendendo un punto e virgola seguito da caratteri a piacere; un'eventuale query string va posizionata dopo tali caratteri, iniziando, come al solito, con un punto interrogativo
  - ovvero, usando una espressione regolare estesa, la parte di un URL che segue l'indirizzo della macchina e del file al suo interno può essere fatto così: `(;[^?]*)?(\?.*)?`
  - ok, non proprio tutti i caratteri sono ammessi, ma è per rendere l'idea
  - normalmente, la parte tra il punto e virgola (compreso) e l'eventuale punto interrogativo (escluso) viene ignorata del Web server
  - tuttavia, le servlet possono richiedere l'indirizzo completo con cui sono state invocate e sfruttare queste informazioni (dietro le quinte)
- Esempio `slide159_urlrewriting`: come si possono sostituire ai cookie le sessioni, tenendo conto del fatto che i cookie possono essere disabilitati
  - da notare che l'HTML statico `slide159.html` usato in precedenza deve diventare dinamico: è necessario riscrivere l'URL nella `action` del `form`...
  - c'è ancora un HTML statico `slide159_urlrewriting.html` da usare all'inizio; ma, per l'appunto, può essere usato solo all'inizio (quando la sessione viene creata) e non più in qualsiasi momento
  - per farlo, il metodo `doGet` è stato programmato con 2 modalità distinte: se la servlet viene invocata con una querystring che contiene `form`, allora ricrea dinamicamente l'HTML statico `slide159_urlrewriting.html`, con l'unico cambiamento del valore dell'attributo `action` del tag `form`, dove appunto viene usato l'URL rewriting
  - se invece non c'è una querystring (o c'è, ma senza il nome `form`), allora il funzionamento è lo stesso di prima (mostra le scelte fatte fino ad ora)
  - tutti i link creati nelle pagine vengono prima passati a `response.encodeURL`
  - provare a disabilitare i cookies sul browser, e verificare che `slide159` non funziona più correttamente, mentre `slide159_urlrewriting` sì