Due o tre cose sulle pagine web – #loptis Reply

#loptis • Tags: AJAX, HTML, Javascript, MySQL, PHP, SQL

Devo consultare l'orario, anzi devo anche fare il biglietto. Mi siedo al computer e chiamo la pagina http://www.trenitalia.it. Devo andare da Firenze a Milano. Scrivo Fi… è già sceso un menu a tendina: "il sistema" mi fa scegliere fra Firenze Campo di Marte, Firenze S. M. Novella, Rho-Fiera Milano, Riminifiera. Furbetto! Ma come fa?

Ragioniamo. Cos'è il sistema? In questo caso il sistema comprende il mio computer, proprio questo su cui sto facendo la prova. Poi il mio router casalingo che lampeggia qui davanti, in sostanza un computerino che sa dove indirizzare i pacchetti di informazione nella rete, il primo nodo che i miei pacchetti di informazione devono superare per inoltrarsi nella rete. Ma i pacchetti ne dovranno superare altri, perché la rete è fatta di nodi collegati fra loro, innumerevoli, e in ognuno di questi possono esserci delle diramazioni. Ho controllato, stasera i miei pacchetti devono superare 15 nodi per raggiungere il server delle ferrovie. Domani forse di più, o di meno. La rete è dinamica: la via per collegare due punti può cambiare. E poi c'è il server delle ferrovie, quello che mi risponde quando chiamo http://www.trenitalia.it. Quest'ultimo è un computerone acceso 24 ore su 24, 7 giorni su 7. In realtà ci sarà un insieme di server, che si suddividono il carico di lavoro con criteri sofisticati. Ma alla fine sarà uno quello che mi risponde.

Un sistema composito e eterogeneo, e dinamico. Fra dieci minuti potrei decidere di fare la stessa cosa con il tablet, o con lo smartphone. Il sistema può essere ricomposto in una varietà di configurazioni fisiche, ma io lo vedo sempre come un'unica "macchina", alla quale chiedo un orario ferroviario.

Io ho la domanda e Trenitalia ha i dati. Io manovro il lato "client", Trenitalia il lato "server". Il client ce l'ho io, il server è lontano. Quando invoco http://www.trenitalia.it, il server delle ferrovie mi risponde, inviandomi una pagina che posso vedere nel browser, su computer, tablet o smartphone che sia. Quella pagina è codificata in HTML, che voi avete già assaggiato. Quando fate clic su uno dei vari link che ci sono nella pagina, io faccio partire una richiesta più specifica al server delle ferrovie. Il server, in base al link che ho cliccato, confeziona una nuova pagina e me la rispedisce. Il mio browser si limita a mostrarmela.

Questo è il meccanismo base dei servizi web. Provatelo, cliccando per esempio il primo link nel secondo menu, subito a destra del logo *TRENITALIA,* nella pagina http://www.trenitalia.it: "Condizioni di trasporto". Vedrete che vi aprirà una nuova pagina, diversa dalla precedente. L'aprirà in un'altra finestra (o scheda, tab, del browser). Questo dipende da come è scritto il link. Se per esempio, andate nel mio blog, http://iamarf.org, e cliccate sul titolo del primo post, #loptis La bacheca, la pagina verrà sostituita da una nuova, che invece di contenere la lista degli incipit dei primi post, conterrà solo quel post con tutti i suoi commenti. Il meccanismo è lo stesso: voi chiedete una nuova pagina e il server ve la invia.

In questo schema, il lavoro fatto dal sistema può essere pensato come diviso in due parti: il lavoro del client, che consiste nel registrare i vostri clic e spedirli al server, e il lavoro del server, che consiste nell'interpretare quei clic, frugare fra i suoi dati, e rispedirvi ciò che avete chiesto.

I dati sono conservati dal server in una cosa che si chiama *database*, o più precisamente *database relazionale*. In questo tipo di database, i dati sono organizzati in tabelle collegate fra loro da un preciso schema di relazioni, il tutto architettato in maniera tale che, anche quando la struttura dei dati diviene molto complessa, la mole ingente e il ritmo delle interrogazioni frenetico, l'integrità dei dati non viene mai messa in pericolo. Esiste un linguaggio concepito apposta per porre interrogazioni a questo tipo di

database, si chiama Structured Query Language (SQL). Esiste anche un'implementazione open source di questo linguaggio che si chiama MySQL. Questa è enormemente diffusa in una miriade di applicazioni in tutto il mondo. Non ci serve sapere altro a riguardo, giusto che esiste.

Ma il lavoro del server non consiste solo nel porre domande al database. Quando il server riceve le informazioni relative ad un click, deve compiere delle elaborazioni che danno luogo a precise interrogazioni da porre al database, quindi deve elaborare le risposte e confezionarle adeguatamente in una nuova pagina HTML da rispedire al vostro client. Questo tipo di lavoro viene realizzato mediante un linguaggio che si chiama Hypertext PreProcessor (PHP). È un vero e proprio linguaggio di programmazione che può essere intercalato in una pagina HTML, utilizzando un particolare tipo di tag, che per la cronaca – ma solo per la cronaca! – è

<?php qui viene la serie di istruzioni scritte in PHP ?>

Il server, prima di rispedire la pagina HTML al client, esegue tutte le istruzioni PHP che ci trova dentro, e che gli consentono di determinare la forma finale con la quale la pagina apparirà nel client.

Non ci serve sapere altro, eccetto mettere a fuoco il fatto che il server per ogni clic che lo "solletica" fa molto lavoro: interpreta le informazioni legate al clic, pone domande ai database, elabora le risposte e confeziona le pagine da rispedire indietro. Se i clic sono molto numerosi, il server può andare in crisi e non farcela.

Questo era il meccanismo su cui si basava il vecchio web, per così dire. Il web pre 2.0, dove le pagine erano dinamiche ma meno di oggi: per cambiare anche un piccolo particolare di una pagina, questa doveva essere rinfrescata completamente. Non è che questo meccanismo non si usi più, tutt'altro, ma solo quando è inevitabile farlo, come nei due esempi precedenti.

In realtà noi oggi siamo abituati a tutto un altro dinamismo delle pagine: la pagina resta là e solo certe sue parti mutano, e in maniera istantanea, non solo in funzione dei clic subiti, ma mediante il semplice "sorvolo" del puntatore del mouse, o anche spontaneamente – anche troppo spesso talvolta. Riprendiamo l'esempio di http://www.trenitalia.it. Se guardate la pagina e aspettate qualche istante vedrete come ogni tanto la pubblicità sottostante cambi. Ma soprattutto, provate a viaggiare con il puntatore del mouse sulle voci "LE FRECCE", "OFFERTE E SERVIZI" eccetera. Vedrete che quando il puntatore si troverà sopra una di queste voci, scenderà istantaneamente un menu a tendina, senza che voi abbiate cliccato.

Chi fa questo lavoro? Il server? No, non potrebbe reagire così velocemente. Il meccanismo che abbiamo visto, richiederebbe che tutta la pagina venisse rinfrescata interamente dopo essere stata spedita, cambiata e resa dal server. Allora il client? Ma come è possibile? Per quanto ne sappiamo, il client non fa altro che spedire clic con qualche informazione e ricevere nuove pagine da mostrare al posto delle vecchie, che abbiamo visto essere codificate in HTML. E se nelle pagine HTML ci sono dentro istruzioni PHP, sappiamo che queste comunque vengono eseguite dal server, prima di essere rispedite indietro. E allora? I nodi intermedi? No, quelli fanno solo un lavoro di bassa manovalanza, loro manco le pagine concepiscono, ma solo piccoli pacchetti di informazioni, nei quali quelle sono spezzettate; i nodi ricevono pacchetti, li smistano e li inviano nelle direzioni che credono giuste. Allora, chi si fa carico di tutti quegli aggiornamenti rapidi e localizzati che siamo abituati a vedere nelle pagine web?

In effetti ci manca un ingrediente. Qualcosa che lavori nel client. E questa cosa c'è: nella maggior parte dei casi si chiama Javascript, ancora un linguaggio. Voi direte – Ma perchè tutti questi linguaggi? – Perché ognuno è adatto a svolgere un certo tipo di attività in un certo contesto. Non è che siano poi molto diversi fra loro, ma sono comunque distinti. Tuttavia lasciamo queste valutazioni a chi si occupa di software. A noi interessa sapere che in una una pagina HTML si possono includere anche sezioni

scritte in Javascript, che vengono eseguite in corrispondenza di certi eventi, per esempio un bottone che viene cliccato o una particolare zona che viene sorvolata.**Queste sezioni di codice Javascript vengono eseguite nel client**, a differenza di **quelle in PHP che vengono eseguite nel server**. Sempre per la cronaca, le sezioni di codice Javascript sono incluse in tag di questo tipo

```
<script>
qui viene la serie di istruzioni in Javascript
</script>
```

Il fatto che il codice Javascript venga eseguito nel client spiega la reattività immediata delle pagine web; per verificare riprovate a volare sulle voci di menu di http://www.trenitalia.it. Ma non è solo questione di accorgimenti grafici. Torniamo a fare la prenotazione con la quale avevamo iniziato: scrivo Fi… è già sceso un menu a tendina: ho appena iniziato a scrivere il nome della città, Fi, e "il sistema" mi fa scegliere fra Firenze Campo di Marte, Firenze S. M. Novella, Rho-Fiera Milano, Riminifiera. Furbetto! Ma come fa?

Beh, ora forse è un po' più chiaro. Evidentemente il codice HTML di quella pagina include una sezione di codice Javascript che si attiva quando inizio a scrivere nella casella "**Da:**". Ma c'è ancora una domanda: per sapere tutte le stazioni che iniziano con i caratteri "Fi", lui deve interrogare il database e questo si trova nel server… allora come fa un codice che gira nel mio client a interrogare il server? In effetti, con le prime versioni di Javascript non sarebbe stato possibile fare una cosa del genere. Poi, dopo alcuni anni è stata sviluppata una tecnica che, mediante codice javascript, consente di porre interrogazioni al database localizzato remotamente, riaggiornando solo alcune parti della pagina al volo, in funzione delle risposte ricevute. Questa "nuova" tecnica si chiama Asynchronous JavaScript and XML (AJAX).

Non entriamo in ulteriori particolari e dimenticate serenamente ciò che vi opprime. Badate però di essere consapevoli del fatto che oggi quando si lavora con una pagina web accadono molte cose nel proprio client il quale, ricordiamo, può essere il computer, il tablet, lo smartphone e tutte le possibili variabili, presenti e prossime venture. La cosa ha due conseguenze rilevanti.

La prima è che grazie a questa stratificazione successiva di tecnologie, il web ha potuto dinamizzarsi enormemente, grazie a una distribuzione più equilibrata del carico di lavoro fra congegni richiedenti (client) e prestatori di servizi (server). In questo modo la stratificazione dei linguaggi e la fluidificazione del codice distribuito in una miriade di macchine, che collegate fra loro formano la rete, hanno reso possibile il web 2.0.

La seconda conseguenza consiste nel fatto che nei nostri congegni succedono molte cose. Sempre più, quando carichiamo pagine web, carichiamo anche un bel po' di software. E il Javascript non è il solo tipo di codice "straniero" che può girare nel vostro computer, ad esempio Java e Flash sono altri tipi di software che possono essere importati con le pagine web. I video Youtube funzionano in Flash per esempio. In generale, tutto ciò si traduce in maggiorate e migliorate funzionalità ma, come sempre succede, anche in qualche problema:

1. Qualsiasi software assorbe risorse dal computer, anche quello che viene con le pagine web. Navigare il web non è assimilabile ad una pratica omogenea di recupero e visualizzazione di documenti; può accadere che il computer rallenti improvvisamente in seguito al caricamento di una pagina web o ad un clic su un particolare comando. Un clic apparentemente innocente può implicare l'attivazione di un software che per una varietà di motivi può risultare particolarmente oneroso.
2. Il software può contenere errori e le conseguenze degli errori in un software sono del tutto imprevedibili: siamo nel regno della complessità e un piccolo difetto può tradursi in una grande varietà di comportamenti anomali.

3. L'avvicendarsi frenetico delle nuove revisioni di tutti i componenti che oggi concorrono ad ogni singola funzionalità del web crea non poca farragine. Ogni volta che un singolo componente evolve in una nuova versione, vari altri componenti devono in qualche maniera adeguarsi. Per esempio, quando esce una nuova versione di HTML, è necessario aggiornare tutti i browser in maniera che siano in grado di interpretare le direttive del nuovo standard. Purtroppo questo processo passa sopra la testa dei poveri utenti, i quali sono così costretti a prendere sul serio le svariate richieste di aggiornamento che periodicamente vengono loro proposte: Browser, Java, Javascript, Flash, sistema operativo eccetera. Ma il problema non concerne solo l'evoluzione temporale. Chi usa contemporaneamente più accessi a internet può incorrere frequentemente in episodi irritanti. Il caso di chi scrive è estremo, per ovvi motivi professionali: se mi rivolgo ad una platea generalizzata non posso ignorare tutte le istanze di coloro che hanno sistemi diversi dal mio! Ecco che non mi succede quasi mai di essere perfettamente a posto con Windows, Mac OSX, Linux, IOS (iPad, iPhone…), Android (Samsung…). Se per esempio ho bisogno di utilizzare PiratePad, ecco che scopro che sull'iPad funziona poco e nulla – se utilizzo un certo tipo di wiki (siti per scrittura cooperativa) ecco che dopo avervi accumulato centinaia di studenti come utenti, scopro che con iPad non funziona affatto – in questo momento ho Flash aggiornato su Windows e su Mac ma su Linux no – eccetera eccetera…
4. Una poco simpatica conseguenza dei fatti precedenti è la rapida obsoloscenza dell'hardware, dal computer allo smartphone. Navigare il web con computer vecchi può rivelarsi un'esperienza defatigante. Non bisogna dimenticare che pare valga ancora la legge di Moore, che semplificando un po', dice: la potenza dei computer raddoppia ogni 1.5 anni. Questo significa che in un lustro le potenzialità dell'hardware sono quasi 10 volte superiori, un ordine di grandezza! Dal canto suo, il software si evolve adeguandosi all'hardware. Ma dal punto di vista dell'utente, aggiornare il software è spesso facile e gratuito, sostituire l'hardware è faticoso e costoso.
5. Il software può essere anche scritto con l'intenzione di fare danno. Una parte del cosiddetto *malware* può giungere attraverso il codice portato dalle pagine web. Niente panico. Il mondo è pericoloso, anche se ce lo raccontiamo sicuro. È il prezzo della vita. Il cyberspazio, ormai duale del reale a tutto tondo, non fa eccezione. Ci si può vivere dentro con un minimo di buon senso, e la consapevolezza che è possibile inciampare in qualche ostacolo. Il buon senso nella fattispecie consiste nell'abitarlo con accortezza, elevando il livello di attenzione nei luoghi sconosciuti. Ad esempio, pagine eccessivamente policromi, con banner mobili, un po' tipo slot machine, siti che consentono di scaricare gratuitamente ma che **non dichiarano esplicitamente di offrire software open o free source**, pagine che offrono sistemi per crackare chiavi o altro software, sono in generale da evitare. Insomma, il cyberspazio non è il paese dei balocchi ma è il mondo, che è pieno di ciarpame e ricchezza allo stesso tempo. Per percorrelo e conoscerlo occorrono attenzione e ponderazione.