Verso Web 2: dalle pagine alle applicazioni

Impiego di Ajax per migliorare l'esperienza dell'utente nell'uso della banca dati di "Biblioteche oggi"

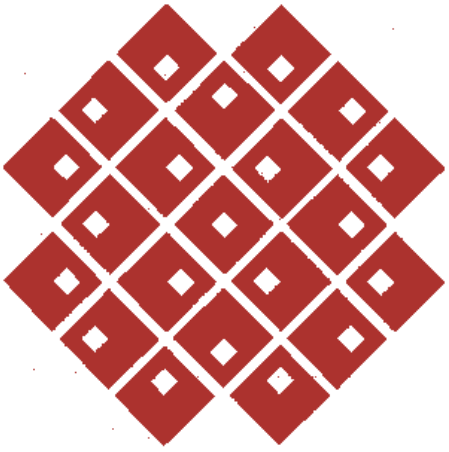
Piero Cavaleri

Biblioteca "Mario Rostoni"
Università Carlo Cattaneo-LIUC
Castellanza
PCavaleri@liuc.it

Evoluzione delle interfacce di OPAC e banche dati bibliografiche

Le interfacce delle banche dati e degli OPAC negli ultimi tredici anni hanno assunto una configurazione fortemente dipendente dal modello di presentazione delle informazioni caratteristico del web.
Il web si propone come un sistema per la costruzione di ipertesti in rete basato su file/pagine autonomi che vengono richiesti da parte di un client ad un server.[1]
In questo modello, compito del browser è richiedere un file e presentarlo all'utente sotto forma di pagina. Il server memorizza nel suo file system i singoli file, e li invia quando vengono richiesti. I file possono essere anche prodotti dinamicamente da un programma di backend, sulla base di una domanda dell'utente corredata di parametri.
La modalità di richiesta di un file consiste, nella gran parte dei casi, nell'attivazione di un elemento speciale del testo visualizzato dal browser, chiamato "link". Richieste più complesse vengono attivate attraverso l'uso di form che l'utente completa con i parametri necessari prima di attivare un elemento deputato all'invio.
Questa modalità di elaborazione dei dati differisce in modo radicale da quella prevalente per le interfacce delle banche dati e degli OPAC dell'inizio degli anni Novanta. Fino ad allora l'interazione tra l'utente e la banca dati avveniva in base a programmi in cui la gestione dell'interfaccia utente risultava fortemente integrata nei processi di interrogazione e di elaborazione dei dati. Le informazioni che comparivano a video erano aggiornate in base alle richieste dell'utente senza che ci fosse la necessità di eliminare l'intera schermata per sostituirla con una nuova. Il programma o manteneva un controllo completo dello stato dell'applicazione, oppure, nel caso di applicazioni client/server, manteneva la connessione in maniera permanente, così da poter reagire in modo continuo alle modifiche apportate dall'utente. Il modello era quello di un'interfaccia che veniva modificata dal programma, che ne controllava lo stato in modo continuo e puntuale, senza mai essere interamente sostituita. Nel caso di programmi client/server non basati sul protocollo http l'interfaccia era ed è sotto il controllo del programma client, che gestisce un collegamento permanente con il programma server.
I produttori di banche dati e di OPAC hanno prima resistito all'introduzione del web, considerando che molte caratteristiche peculiari delle loro realizzazioni dovevano essere abbandonate per i limiti imposti dal nuovo modello. In seguito alla diffusione di Internet hanno aggiunto interfacce adatte a questo modello. Infine la grande diffusione del web li ha convinti ad abbandonare le interfacce proprietarie per affidarsi esclusivamente al browser come agente per produrre l'interfaccia rivolta all'utente finale. Nel fare ciò le aziende produttrici hanno dovuto affrontare non pochi problemi e costi, con risultati solo in parte soddisfacenti. Infatti, oltre al limite fondamentale di essere *connection-less*, fatto cui ogni produttore di siti cerca di ovviare per mezzo di sessioni e cookie, il modello per pagine del web presenta degli svantaggi nel momento in cui è necessario svolgere un'attività di ricerca complessa e prolungata, come nel caso della ricerca documentale o bibliografica.
Il primo problema che emerge è di carattere generale, cioè prescinde dal contesto della ricerca documentale. Ogni volta che si richiede un'informazione aggiuntiva, server, browser e infrastrutture di telecomunicazione devono rielaborare anche i dati relativi alle parti della pagina che non devono essere modificate.

Il secondo difetto è dovuto alla difficoltà di gestire la "storia" della sessione di ricerca. Il browser, infatti, mantiene i dati relativi alle varie pagine che mostra in modo separato e indipendente. Questi dati possono essere richiamati solo come un tutto unico attraverso l'uso dei pulsanti "avanti" e "indietro" del browser stesso, oppure nei browser più recenti con l'ausilio della navigazione a schede (*tabbed browsing*). Non sono possibili ulteriori rielaborazioni dei dati memorizzati, così come è difficoltoso eliminare dalla "storia" tutte le pagine intermedie necessarie solo per l'inserimento dei parametri di ricerca.
Il terzo svantaggio risiede nella complessità del controllo del comportamento dell'utente attraverso le molteplici schermate necessarie per visualizzare nei modi adeguati i diversi formati di dati.
Un quarto difetto del modello per pagine deriva dall'estrema difficoltà di introdurre un meccanismo efficace per consentire all'utente di salvare informazioni bibliografiche di suo interesse nel corso di una sessione di ricerca.
L'evoluzione dei browser e degli standard tuttavia consente, da qualche anno, di intraprendere nuove strade per la gestione delle interfacce di banche dati e OPAC. Questa possibilità ha preso il nome di Ajax.[2]
Ajax non è in sé una nuova tecnologia, ma l'applicazione di un insieme di tecnologie già esistenti,[3] collegate sulla base di alcune idee innovative riguardo a come dovrebbe svolgersi l'interazione tra server, browser e utente.
Il suo impiego è ormai consolidato. In particolare l'uso molto esteso che alcuni dei grandi attori di Internet, Google *in primis*, fanno di Ajax per servizi destinati a milioni di utenti consente di affermare con sicurezza che la fase di sperimentazione può essere considerata superata.



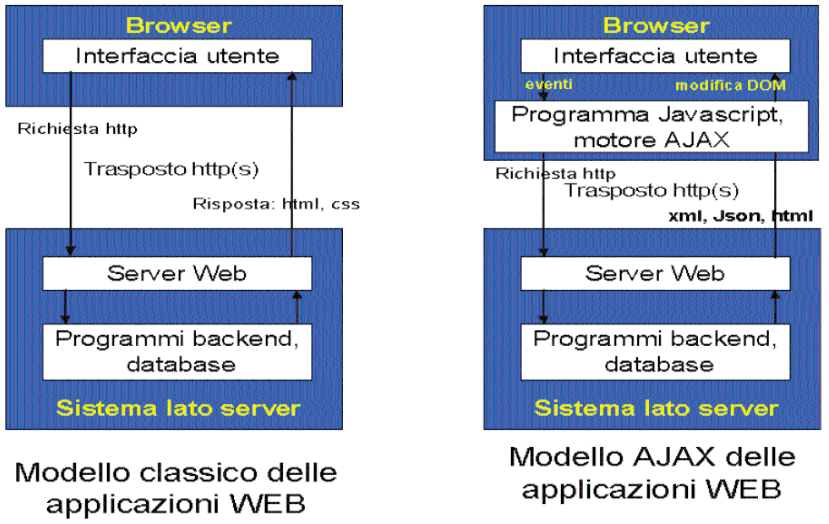
> Google has done more than any other company to raise the profile of Ajax pubblications (and it, like the majority of adopters, was doing so before the name Ajax was coined). Its Gmail service was launched in beta form in early 2004. [...] the main buzz around Gmail was the UI, which allowed users to open several mail message at once and which updated mailbox list automatically, even while the user was typing in a message. [...] Google Maps is a cross between a map viewer and a search engine. [...] The search feature function as a classic web app, refreshing the entire page, but the map itself is powered by Ajax.[4]

Descrivere che cosa si può fare con Ajax è più complicato che utilizzarlo. Infatti, essendo un insieme di tecnologie, consente sviluppi diversi a seconda dell'aspetto su cui una specifica applicazione pone maggiormente l'enfasi.

La tecnologia Ajax

Ajax, come spesso capita con le tecnologie informatiche, è un acronimo, e sta per "Asynchronous JavaScript and XML". In realtà il nome rende poco l'idea di che cosa è in realtà. In particolare pur corrispondendo la X di Ajax a XML, non vi è una dipendenza dall'utilizzo di tale formato nella trasmissione di dati tra client e server perché un'applicazione possa essere considerata appartenente a questa tipologia. Si possono realizzare applicazioni con Ajax senza utilizzare minimamente XML stesso.[5]
Le caratteristiche indispensabili per parlare di impiego di tecnologie Ajax sono l'interazione asincrona in background tra browser e server, e l'uso di JavaScript[6] per gestire tali interazioni e le conseguenti modifiche alla pagina visualizzata. Per il trasferimento dei dati tra i server e il browser si possono utilizzare i formati XML, HTML o Json, a seconda delle esigenze.
Ajax è una parte rilevante di ciò che viene identificato con il termine Web 2. Infatti è l'elemento che consente di realizzare delle applicazioni che offrono all'utente un'esperienza d'uso più ricca, perché le informazioni visualizzate possono essere modificate continuamente senza bisogno di cambiare l'intera pagina. L'aggiornamento delle informazioni diviene "fluido", la risposta alle azioni dell'utente diretta. Invece, rispetto a una delle altre caratteristiche che viene attribuita a Web 2, il rilievo degli stili e dei comportamenti comunitari, partecipativi, Ajax si dimostra neutro.
La caratteristica più rilevante di Ajax rispetto al normale flusso delle informazioni del web sta nella gestione dell'interazione con il server in background rispetto all'interfaccia utente.
Il funzionamento di questa modalità è ben esemplificato negli schemi riportati nelle figure 1 e 2.
L'altra caratteristica fondamentale di Ajax è l'aumento del ruolo del client nelle applicazioni web. Con Ajax queste applicazioni assumono un aspetto client/server più completo rispetto al semplice ruolo di visualizzatore di file di testi che il browser ha nel modello web classico.
L'impiego esteso di JavaScript per gestire i dati che browser e server

Fig. 1 – Modello classico e modello Ajax delle applicazioni web



Rielaborazione dell'autore sul modello presentato in JESSE JAMES GARRET, *Ajax: a new approach to web applications*, <http://www.adaptivepath.com/ideas/essays/archives/ 000385.php>.

si scambiano facilita l'introduzione di ulteriori elementi di DHTML. Lo spostamento del controllo dell'aspetto della pagina a programmi di *scripting* fa sì che divenga spontaneo ampliare l'uso degli stessi programmi al fine di rendere più interattiva, dinamica e facilmente utilizzabile l'interfaccia stessa.
Allo stesso modo, pur non essendo indispensabile, Ajax favorisce un approccio strutturato ai dati scambiati tra server e browser con l'impiego di XML o di Json.
Queste tecnologie possono offrire molte opportunità, che cercheremo di dimostrare, ma richiedono che l'utente disponga di browser in grado di supportare JavaScript, l'oggetto XMLHttpRequest e che applichino in modo il più possibile rigoroso gli standard del W3C.[7]
Le versioni dei principali browser rilasciate negli ultimi cinque anni offrono caratteristiche compatibili con tali esigenze. L'oggetto XMLHttpRequest deve essere gestito con modalità diverse nei browser basati su Gecko/Mozilla rispetto a Microsoft Explorer, ma, sia utilizzando propri script che impiegando librerie ormai collaudate, questo problema è facilmente gestibile.

Vantaggi di Ajax per le applicazioni bibliografiche

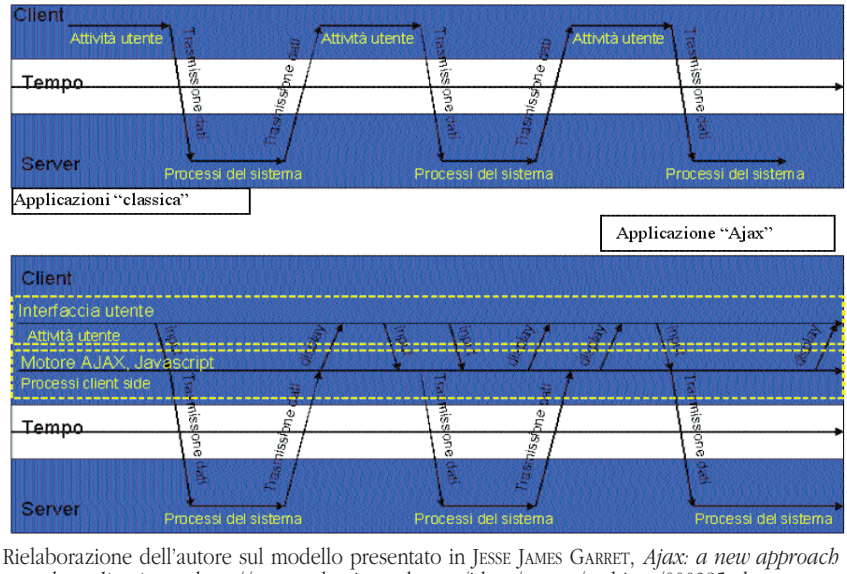
Quali possono essere i vantaggi delle tecnologie Ajax per le interfacce delle banche dati e per gli OPAC?
Per prima cosa l'utente può, finalmente, tornare a integrare informazioni già ottenute con tutte le aggiunte disponibili sul server senza dover ricaricare la pagina.
Per chi gestisce il programma, tale possibilità elimina la necessità, molto onerosa, di tenere traccia di come sia configurata la pagina di partenza dell'utente per poter proporre l'informazione aggiuntiva richiesta senza alcuna perdita nelle informazioni già disponibili. Questo compito, infatti, è trasferito al browser che ha il vantaggio di conoscere esattamente lo stato della pagina in ogni istante.
Un secondo importante vantaggio è la velocità: sono richiesti, elaborati e trasferiti esclusivamente i dati indispensabili a produrre le informazioni richieste, nulla di ciò che l'utente ha già a disposizione viene ritrasmesso. Attraverso l'impiego di formati compatti come Json si può arrivare a ridurre del 90% i dati trasmessi. Una riduzione percentualmente minore, ma sempre significativa, può essere ottenuta riguardo al tempo richiesto per l'elaborazione dei dati da parte dei programmi di backend del server web.
In terzo luogo si può ottenere una notevole semplificazione della gestione del salvataggio delle informazioni che l'utente vuole conservare. L'utente può accumulare i dati di suo interesse attraverso una sessione composta da molte transazioni, per salvarli solo quando ritiene utile farlo.
Un quarto punto a favore di queste tecnologie è la dinamicità dell'interfaccia. Tale caratteristica consente di presentare su una singola pagina un numero molto maggiore di possibilità di interazione browser/server rispetto alle pagine di ricerca di banche dati e OPAC basati sul modello web a pagine.
Infine, un potenziale beneficio, che dipende in gran parte da come evolverà il modo in cui i browser gestiscono la sicurezza, potrebbe venire dalla possibilità di integrare, in modo semplice e basato su standard, informazioni provenienti da server diversi.[8]
In sostanza, con Ajax si possono realizzare Single Page Application, cioè applicazioni, anche complesse, che utilizzano un'unica pagina web, che viene modificata in base alle più varie richieste dell'utente. I dati possono essere ottenuti dal server di origine dell'applicazione oppure, per via indiretta, anche da fonti esterne.

L'interfaccia della banca dati di "Biblioteche oggi"

La banca dati di "Biblioteche oggi" ci è parsa un caso interessante e, nello stesso tempo, non eccessivamente complesso per sperimentare le possibilità offerte dalle tecnologie Ajax. In realtà già dalla fine del 2005 alcuni impieghi di Ajax

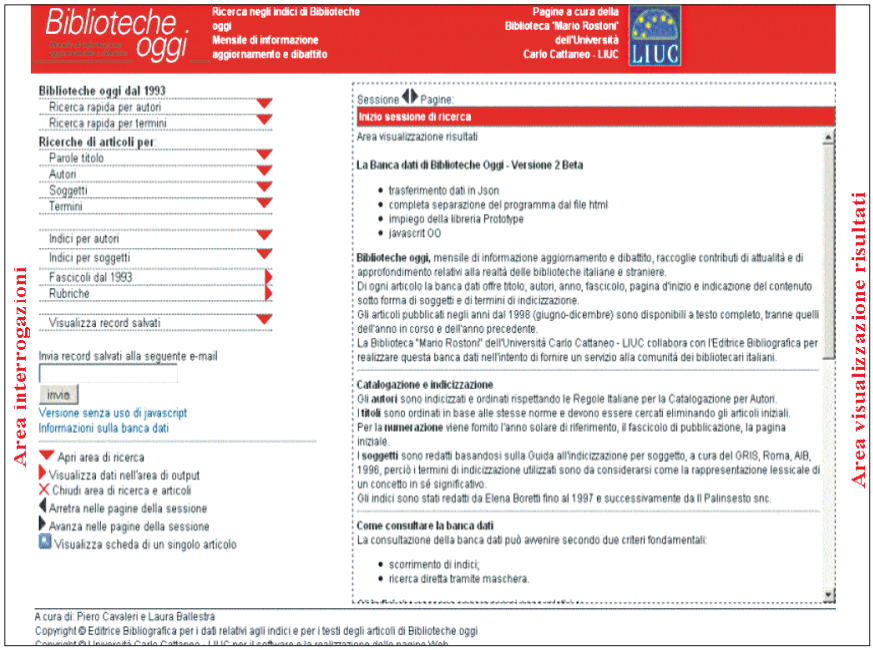
erano stati introdotti nell'interfaccia di questa banca dati: la ricerca veloce degli autori e la visualizzazione contestuale dei dettagli di un articolo. Ora però si è voluto sperimentare la realizzazione di un'applicazione che si svolga interamente in un'unica pagina web, con la gestione di tutte le interazioni con il server web svolte in background e la riduzione al minimo dell'elaborazione del server e della quantità di dati trasmessi.
Premessa necessaria alla spiegazione di come funziona l'interfaccia basata su Ajax è la presentazione delle caratteristiche del database impiegato per gestire i dati e del software lato server impiegato.
La banca dati di "Biblioteche oggi", distinta dai full text presso il server dell'Editrice Bibliografica, risiede sui server della Biblioteca "Mario Rostoni" dell'Università "Carlo Cattaneo" di Castellanza.
Questi server utilizzano il sistema operativo Windows e impiegano il server Web Microsoft Internet Information Service. Il server Web Apache è utilizzato solo per scopi di benchmark. La base di dati che contiene i dati bibliografici è stata sviluppata impiegando il DBMS Microsoft Access. In seguito è stato effettuato il porting su MySQL, che ora viene utilizzato in produzione.
Il software che gestisce la banca dati sul lato server è realizzato sia nel linguaggio ASP – versione in produzione – che nel linguaggio PHP – in versione di test. I due programmi utilizzano rispettivamente MIIS e Apache.
La scelta, fatta nel solco della tradizione della Biblioteca "Rostoni", è stata di non vincolarsi ad alcuna tecnologia, di usare ogni strumento per ciò che di meglio possa offrire nelle condizioni date. Le disponibilità di strumenti software e hardware sono state considerate come dei vincoli, mentre si è cercato di espandere il più possibile le competenze, il know-how relativo alla gestione del nuovo modello d'interfaccia e all'interazione client/server.
Fatte queste premesse passiamo a descrivere come funziona la nuova interfaccia della banca dati, che utilizza come linguaggio di programmazione lato client esclusivamente JavaScript.
La pagina web che gestisce l'interazione tra utente e server appare in due versioni a seconda della pagina da cui si origina la chiamata. Questo è dovuto ad un accordo tra l'editore di "Biblioteche oggi" e la Biblioteca "Rostoni", che prevede una diversa disposizione degli elementi identificativi se gli utenti accedono dal sito di "Biblioteche oggi" oppure direttamente dalla Biblioteca "Rostoni".
Questa doppia presentazione non influenza in alcun modo il funzionamento dell'interfaccia: in ognuno dei casi tutta l'interazione avviene senza cambiare la pagina.
Rispetto alla precedente versione a più pagine, la gestione di questo aspetto è divenuta molto più agevole, perché non vi è più la necessità di tener conto, durante la sessione di ricerca, di quale sia l'aspetto complessivo da mantenere, visto che esso viene definito una sola volta al momento della prima visualizzazione della pagina stessa.
Per descrivere l'applicazione utilizzeremo come modello la pagina priva della contestualizzazione nella Biblioteca "Rostoni" perché più semplice, dato che non contiene tutte una serie di elementi utili esclusivamente per la navigazione nel sito della biblioteca stessa e dell'Università "Cattaneo".
Al momento del caricamento la pagina presenta a sinistra un'area dove sono disponibili tutte le modalità di interrogazione e a destra un'area contenente la presentazione della banca dati, nella quale verranno collocati tutti i risultati delle interrogazioni stesse (figura 3).
Le interrogazioni disponibili sono presentate in modo unitario. Alcune sono immediatamente eseguibili, mentre altre richiedono che l'utente apra un'apposita finestra in cui compaiono, di volta in volta, elementi in grado di attivare una specifica richiesta (nel modello a pagine li chiameremmo "link") o i form da compilare per eseguì-

Fig. 2 - Dinamica temporale del flusso delle attività nelle applicazioni classiche e Ajax



Rielaborazione dell'autore sul modello presentato in Jesse James Garret, *Ajax: a new approach to web applications*, <http://www.adaptivepath.com/ideas/essays/archives/000385.php>.

Fig. 3 – Interfaccia utente della banca dati di "Biblioteche oggi"



re richieste basate su parametri. La distinzione tra interrogazioni eseguibili e comandi per l'apertura delle finestre è realizzata attraverso un diverso segno grafico: un triangolo rosso con la punta a destra nel primo caso, con la punta verso il basso nel secondo.

Ogni finestra può essere chiusa e riaperta tutte le volte che l'utente lo riterrà necessario. Si è scelto di chiudere automaticamente la finestra aperta al momento della richiesta di apertura di un'altra, per evitare che l'aspetto della pagina divenisse poco chiaro per la presenza di troppe finestre aperte.

Le interrogazioni possono essere di due tipi: ricerca di titoli, in base a vari criteri, oppure scelta all'interno degli authority file degli autori e dei termini di indicizzazione. Ogni interrogazione produce un cambiamento immediato nell'area di destra. L'utente viene avvertito con una scritta, "Caricamento in corso...", e con il cambio del colore di fondo dell'attività che il browser sta svolgendo. Questi segnali sono percepibili solo quando il tempo di attesa è superiore a qualche decimo di secondo.

Le informazioni che il browser riceve dal server vengono immediatamente elaborate e visualizzate, oltre ad essere memorizzate per poter essere richiamate in ogni momento attraverso le funzioni di navigazione nella sessione. Dal processo di memorizzazione sono state escluse la ricerca sugli authority file, a causa della loro natura dinamica e delle finalità di selezione piuttosto che di consultazione di informazioni.

La navigazione nella sessione può avvenire attraverso due pulsanti – triangoli neri rivolti a sinistra e a destra – che consentono l'avanzamento o l'arretramento di una pagina alla volta – oppure scegliendo direttamente una pagina in base al suo numero identificativo.

Gli elenchi di titoli presentano ordinamenti dipendenti dal criterio di ricerca adottato. Per esempio, ricerche per parole del titolo e termini offrono un ordinamento basato sui titoli. La ricerca per fascicolo, invece, dà luogo a un elenco ordinato in base alla paginazione. Le rubriche sono ordinate cronologicamente, tranne "Biblioteca professionale" che offre anche l'ordinamento per soggetti. Questa eccezione dipende dal fatto che i soggetti delle recensioni sono strutturati con in prima posizione il cognome e il nome di ognuno degli autori della pubblicazione recensita. Il risultato è un elenco per autori di tutte le opere recensite su "Biblioteche oggi" dal 1993.

La ricerca sugli authority file adotta una tecnica di attivazione basata semplicemente sulla scrittura delle lettere (*type-ahead*) in una casella di un form. Non appena l'utente immette una chiave di ricerca di tre caratteri, questa viene inviata al server come chiave per un'interrogazione. Man mano che si inseriscono ulteriori caratteri, vengono selezionati solo gli elementi dell'elenco che corrispondono alla chiave inserita. Nell'area di destra della pagina vengono mostrati i risultati ottenuti per ogni ulteriore inserimento. Questo tipo di ricerca, tipico della tecnologia Ajax, è spesso impiegata per proporre elenchi di suggerimento che compaiono automaticamente mentre si scrive la chiave di ricerca.

Caratteristiche comuni a tutti gli elenchi di titoli è quella di fornire immediatamente solo le informazioni essenziali. A seconda delle versioni del programma, di cui si dirà più avanti, vengono mostrati tutti i titoli oppure solo i primi cinquanta, con la possibilità di visualizzare volta per volta i successivi cinquanta, fino al termine dell'elenco.

Negli elenchi, per ogni articolo vengono mostrati il titolo e i riferimenti a numerazione e paginazione. Le informazioni complete possono essere visualizzate attraverso una richiesta in background che acquisisce i dati necessari a popolare una nuova finestra contenente, oltre a titolo e numerazione, anche gli autori, i soggetti, i termini di soggetto e l'eventuale collegamento al full text. Questi elementi sono proposti come punti di attivazione per ulteriori ricerche.

L'ultimo elemento di ogni scheda

di articolo è un comando speciale, "Salva", che consente di archiviare il record in un'area visualizzabile nella parte bassa del lato sinistro della schermata.
Quest'area ha la funzione di conservare tutte le notizie che l'utente ritenga utili per un uso futuro. Il contenuto di quest'area può essere inviato via e-mail a una casella postale indicata dall'utente stesso di volta in volta. Le alternative a tale metodo – archiviazione sul server o invio a una casella di e-mail predefinita – comportano la necessità di richiedere all'utente di registrarsi, operazione sicuramente interessante per mantenere i contatti con i propri utenti ma che, notoriamente, scoraggia un numero consistente di utilizzatori.
Ogni integrazione di un elenco con informazioni dettagliate relative ad articoli viene riportata anche nella versione memorizzata dello stesso, in modo che la navigazione nella sessione proponga la sua ultima versione.
Per quanto riguarda la stampa, con i browser più utilizzati – Internet Explorer dal 6 in avanti e Firefox/Mozilla – è possibile riprodurre la pagina come appare nel momento in cui viene richiesta la stampa, cioè con tutte le aggiunte che sono state prodotte attraverso la visualizzazione di nuove informazioni ottenute tramite accesso in background al server. Sarà stampata solo un'intestazione che riporta i dati dell'archivio e l'area di output, in modo da riprodurre esclusivamente le informazioni utili, tralasciando gli elementi funzionali alla ricerca. Questa modalità è realizzata attraverso l'uso del CSS.

I programmi

La realizzazione della nuova applicazione per l'interrogazione degli archivi di "Biblioteche oggi" è stata intesa come una vera e propria attività di ricerca, sperimentazione e apprendimento e, sulla base di tali intendimenti, il lavoro è consistito in gran parte nella realizzazione e nel confronto di varie alternative per valutare i vantaggi, le difficoltà, le differenze sostanziali e marginali.
La prima versione messa in rete è stata quella che corrisponde maggiormente alle capacità pregresse della Biblioteca "Rostoni" per quanto riguarda la programmazione lato server, quella lato client e il formato di trasporto dei dati. In particolare lo stile di programmazione adottato per JavaScript è quello tradizionale sequenziale, con progressivo aumento dell'impiego della programmazione ad oggetti. Il formato dei dati trasmessi è stato inizialmente il semplice HTML, riuscendo in tal modo a utilizzare tutti i programmi di output utilizzati dalla precedente versione della banca dati.
Questa versione non usa librerie di programmi di terze parti e tende a minimizzare le dimensioni degli script in JavaScript. L'intenzione principale è di fornire all'utente un'esperienza d'uso migliore di quella tradizionale senza richiedere un lavoro particolarmente oneroso al browser.
Il carico del server e della rete sono stati solo di poco ridotti rispetto a un'applicazione a pagine.
Una volta realizzata questa versione iniziale, solo parzialmente innovativa, si sarebbero potute sperimentare molte combinazioni diverse delle tecnologie Ajax e non, impiegando nelle varie parti dell'applicazione software e tecniche diverse. Per esempio, si sarebbe potuto cambiare il database utilizzato, il programma di produzione dei dati sul server, il linguaggio dei programmi lato server, il server web stesso, il sistema operativo del server, i programmi lato cliente, la struttura della pagina web e del CSS.
La maggior parte di queste alternative non avrebbe però rappre-

sentato una reale novità per quanto riguarda le modalità di svolgimento del flusso di lavoro.
L'unico settore dove si è ritenuto interessante sperimentare ipotesi nuove era quello della massimizzazione dell'impiego del browser con la contemporanea minimizzazione del lavoro del server e il maggior contenimento possibile della quantità di dati trasferiti.
Si è perciò realizzata un'altra versione, messa in linea nella versione beta, che intende perseguire proprio tali scopi. In questa versione, il carico maggiore del lavoro è stato attribuito al browser, cioè a un programma in JavaScript che si avvale anche dei metodi messi a disposizione dalla libreria Prototype,[9] mentre i dati trasmessi dal server al browser sono codificati nel formato Json (JavaScript Object Notation).
La scelta di impiegare la libreria di metodi Prototype rappresenta un cambiamento molto rilevante rispetto alla prima versione. In rete si possono trovare varie ottime librerie che offrono metodi molto semplici per la gestione del flusso di dati tra browser e server, ma in un primo tempo si è scelto di non impiegarne alcuna, perché nell'applicazione si prevedeva di utilizzare solo poche funzioni a fronte delle molte messe a disposizione da queste librerie, che perciò sarebbero state sovradimensionate rispetto alle esigenze. Il programma JavaScript che gestisce il colloquio con il server nella prima versione richiede solo pochi KB contro le decine di queste librerie.
Indubbiamente, la seconda versione, realizzata utilizzando Prototype, consente una gestione più semplice delle richieste e di sfruttare funzioni molto potenti, ma il prezzo da pagare è il caricamento di programmi che complessivamente ammontano a circa 100 KByte.
Sul lato server la seconda versione del programma, realizzata sia in Vbscript che in PHP, risulta molto più semplice perché i dati sono confezionati in formato Json che richiede solo pochi elementi strutturali, rispetto alla prima versione che invece processa i dati richiesti in modo da confezionarli già in formato HTML, immediatamente utilizzabile per modificare gli elementi della pagina di destinazione ma anche molto più oneroso in termini di quantità dei dati trasmessi.
Come già detto, entrambe le scelte, pur escludendo XML, rientrano pienamente nella tecnologia Ajax.
La produzione lato server di documenti HTML già pronti per essere inseriti nella pagina finale, garantisce buone prestazioni anche con browser e pc poco potenti, mentre l'impiego di Json riduce al minimo i tempi di elaborazione del server e il flusso di dati, richiedendo però un oneroso processo di elaborazione dei dati ricevuti dal client. L'elaborazione dei dati Json risulta più efficiente di quella con file XML.
Mentre la prima versione messa in linea può essere considerata priva di particolari difficoltà di realizzazione, la seconda ha richiesto l'esplorazione di varie alternative e un intenso lavoro di verifica dei molti possibili problemi. Essa è quella su cui si dovrà lavorare per le interessanti prospettive di sviluppi ulteriori anche se prima di essere considerata stabile dovrà avvalersi di sperimentazioni che coinvolgano un ampio campione di utenti al fine di consentirne la valutazione in molteplici situazioni.
La scelta di rinunciare a XML, come formato di dati strutturato, a favore di Json, un formato sicuramente meno noto, richiede una giustificazione. A nostro parere, questo formato offre gli stessi vantaggi di interoperabilità e di pubblicità garantiti da XML, ma si basa su una struttura più semplice e in vari ambienti di programmazione, tra i quali JavaScript, può essere processato in modo più diretto e semplice. Il confronto tra HTML, XML e Json può essere approfondito nell'apposita scheda (box a pagina 42).
La struttura dell'applicazione prevede otto tipi di interazioni tra interfaccia utente e programma JavaScript, che a loro volta corrispondono a quattro modi di interazione tra programma e server web.
I modi di interazione interfaccia utente/programma JavaScript sono:
– apertura e chiusura di una finestra di interrogazione;
– richiesta di interrogazione attraverso un link;
– richiesta di interrogazione attraverso un form;
– richiesta di interrogazione attraverso la digitazione di caratteri in un elemento di un form;
– richiesta e visualizzazione di dati relativi a un articolo;
– cambiamento delle informazioni da visualizzare tratte da quelle memorizzate durante la sessione;
– salvataggio nel browser di dati relativi a un articolo;
– invio tramite messaggio di posta dei dati relativi ad articoli salvati.
Le modalità di interazione tra il motore Ajax e il server sono:
– richiesta di dati relativi a liste di articoli con metodo post;
– richiesta di dati relativi a liste di articoli con metodo get;
– richiesta di dati relativi a liste di autorità;
– richiesta di dati relativi a singoli articoli;
– salvataggio di dati di articoli salvati.
Le interazioni tra interfaccia utente e programma JavaScript riguardano sostanzialmente la modifica delle modalità di visualizzazione della pagina, la richiesta di nuove informazioni al server, la modifica delle informazioni da visualizzare, la memorizzazione delle informazioni già visualizzate, il salvataggio delle informazioni di proprio interesse.
L'interazione tra motore Ajax e server corrisponde a tutte le possibili richieste di dati necessari alla modifica delle informazioni già pos-

sedute dal browser e all'invio dei dati salvati per la produzione da parte del server del messaggio di e-mail da inviare alla casella postale dell'utente.
La prima versione, come già detto, presenta pochi aspetti interessanti, perché i dati giungono dal server in modo tale da non consentire scelte particolari. I file HTML possono essere usati solo per modificare l'interfaccia utente.
Nella seconda versione, invece, il formato Json consente di scegliere in quale maniera sia più efficiente elaborare i dati.
Una scelta importante ha riguardato il modo con cui vengono trattati i dati recuperati attraverso interrogazioni dinamiche sulle liste di autorità. Nella prima versione l'inserimento di una lettera nelle caselle previste per l'interrogazione delle liste di autorità degli autori e dei termini produceva innanzi tutto l'attivazione di una procedura per verificare se la stringa inserita fosse o meno più lunga di due caratteri. Nel caso i caratteri fossero più di due, il motore Ajax invia un'interrogazione al server che restituisce l'elenco di nomi o termini inizianti con i caratteri immessi.
Ad ogni successiva immissione si ripete un'interrogazione. Questo metodo è inefficiente e rischioso, infatti le successive interrogazioni non fanno altro che richiedere sottoinsiemi dell'elenco già ottenuto in base ai primi tre caratteri, mentre la rapidità con cui le richieste possono essere inviate può mettere in crisi il server.
Nella seconda versione il fatto di poter disporre di un complesso strutturato di dati consente di eseguire le successive interrogazioni sull'insieme ottenuto in base a tre caratteri, senza più interagire con il server.
In realtà questa tecnica presenta alcuni problemi dovuti alla natura asincrona delle richieste che il motore Ajax fa al server. L'inserimento di ulteriori caratteri dopo il terzo può precedere la ricezione della risposta del server, con il risultato che la selezione non può compiersi per assenza di dati. La gestione di tale problema è particolarmente complessa e soggetta a rischi, tanto che in alcuni siti, per esempio Google, si fa la scelta di rallentare la velocità di inserimento delle lettere in modo da evitare una richiesta di raffinamento prematura. Nel nostro caso si è preferito verificare ad ogni nuovo inserimento la disponibilità dell'insieme su cui fare la selezione. Eventualmente si accoda un'ulteriore richiesta al server. Questa seconda richiesta produce un insieme più ristretto e quindi dovrebbe essere soddisfatta più velocemente. Tale aspetto dell'applicazione è tra quelli che dovrà essere meglio valutato in fase di testing.
Un'altra differenza tra le due versioni riguarda il trattamento degli elenchi di articoli. Nella prima versione il server invia un elenco già formattato e completo di tutti gli elementi necessari per l'acquisizione dinamica delle informazioni dettagliate relative a un articolo. Questi elenchi possono essere di dimensioni consistenti e perciò richiedere tempi non brevi per l'elaborazione sul server, per il trasferimento e la ricezione, ma una volta disponibili possono essere immediatamente visualizzati.
Al contrario i dati ricevuti dal browser nella seconda versione sono strutture Json molto sintetiche, perché contengono poco più dei dati veri e propri, ma una volta ricevuti richiedono una fase elaborativa che può essere non brevissima. La scelta in questo caso è stata quella di convertire in HTML solo i primi cinquanta record, per poi lasciare all'utente la possibilità di convertirne man mano altri cinquanta, sino al completamento dell'elenco, qualsiasi sia la sua lunghezza. Questa scelta comporta la memorizzazione dell'elenco sia come oggetto Java-

Script, sia nel formato HTML. L'oggetto JavaScript viene eliminato solo quando un elenco è stato completamente convertito.
Riguardo invece ai dati completi relativi ai singoli articoli, l'unica differenza tra le due versioni consiste nel fatto che l'onere di produrre il formato HTML viene in un caso addossato al server, nell'altro al client.
Una possibilità, non ancora esplorata perché non si è in grado al momento di valutarne la rilevanza pratica, consiste nella memorizzazione dei dati dei singoli articoli in un *array hashed*, in modo da verificare la disponibilità degli stessi prima di procedere a ogni ulteriore richiesta. L'aggiunta di questo carico di lavoro e l'impiego ulteriore di memoria potranno essere giustificati una volta che si sia quantificata la presenza di più richieste dei dati di un medesimo articolo nel corso di una sessione.
Un'altra possibilità per le liste di titoli, non sperimentata perché le scelte implicate non sono parse interessanti, è quella di caricare tutti i dati relativi agli articoli selezionati in modo da poterli trasformare senza dover fare ulteriori richieste al server. Questa possibilità richiede un'attenta valutazione dei tempi di elaborazione sul server, delle dimensioni dei file di dati trasferiti e della quantità di memoria occupata lato client, poiché la quantità di dati elaborati e trasmessi per ogni interrogazione quadruplicherebbe. Con buona probabilità, per realizzare un'applicazione efficiente bisognerebbe imporre dei limiti alla lunghezza delle liste dei titoli individuati, con evidente perdita di molti vantaggi del modello a liste complete. Questa opzione comunque richiede sicuramente approfondimenti, specie qualora sia necessario imporre limitazioni alla lunghezza delle risposte dovuta al frequente prodursi di elenchi eccessivamente ampi. Nel caso della banca dati di "Biblioteche oggi" questo non può accadere, perché vi sono pochi elenchi che superano i 200 titoli e al massimo raggiungono i 1.000, numeri ben gestibili dal programma e, soprattutto, dagli utenti.
Nella seconda versione, il programma backend può considerarsi come un vero e proprio servizio, dotato di un'interfaccia e di formati di output tali da poter essere utilizzati da applicazioni lato client diverse. Nell'ipotesi di realizzare pienamente questo modello si potrebbe pensare che applicazioni prodotte da altri possano essere autorizzate a interrogare il database per realizzare servizi diversi.
Sul lato client una caratteristica molto importante della seconda versione è la separazione completa del codice HTML dal programma JavaScript. Nella pagina HTML compaiono esclusivamente i comandi per caricare i file dei programmi in JavaScript, mentre la gestione degli eventi è realizzata assegnando questi come metodi degli elementi della pagina stessa.
In tale modo il codice HTML, il programma in JavaScript, il programma di backend sul server e il database vengono separati completamente. Tutto ciò che chi realizza ognuna delle componenti deve conoscere è l'interfaccia delle altre parti con cui la propria deve interagire.
In particolare chi realizza la pagina web dovrà assegnare agli elementi della pagina gli identificativi e le classi CSS necessarie, ma sarà assolutamente libero di disporre le varie parti come gli sembrerà più opportuno, ad eccezione naturalmente di quelle che devono agire in modo coordinato come i pulsanti di apertura e chiusura delle finestre.
Chi realizza le pagine web non deve preoccuparsi di inserire le chiamate alle funzioni di JavaScript necessarie all'interazione dell'interfaccia utente con il programma *client side*.
In questo modello viene meno anche la necessità di utilizzare il tag <a>, perché tutti i link sono sostituibili da eventi sugli oggetti.
Nella nostra realizzazione abbiamo mantenuto l'utilizzo di questo tag perché i browser attuali lo riconoscono come specifico e in base alla sua presenza consentono comportamenti particolari: lo spostamento sulla pagina utilizzando il tabulatore, la visualizzazione di un cursore specifico, modifiche della grafica. Inoltre i programmi di sintesi vocale individuano il tag <a> come un'area particolare, leggendone il valore con l'indicazione che si tratta di un link. Riguardo ai problemi relativi all'accessibilità, tutt'altro che risolti, un accorgimento necessario consiste nell'assegnare sempre per l'attivazione delle funzioni una coppia di eventi: uno relativo al mouse, l'altro alla tastiera.
I problemi più critici riscontrati con un'applicazione come questa sono di due tipi: quelli comuni a tutte le applicazioni Ajax e quelli specifici alle applicazioni bibliografiche.
Per prima cosa, si deve prestare la massima attenzione alla codifica dei caratteri dei messaggi che vengono scambiati tra server e browser. I software coinvolti nel processo sono vari e ciò pone vincoli di vario tipo che devono di volta in volta essere risolti considerando attentamente come trattare i file trasmessi in dipendenza dalle varie parti. Questo aspetto è poco trattato nei manuali in lingua inglese, perché questa lingua non contempla la presenza di caratteri con diacritici.
Un altro problema generale delle applicazioni Ajax riguarda la modifica delle abitudini nell'interazione tra utente e browser. Durante l'impiego di un'applicazione Ajax i tasti "avanti" e "indietro" del browser non sono utilizzabili per navigare attraverso le informazioni già visualizzate. Il modello su cui si

basa la logica dei browser sono le pagine singole immodificabili che vengono memorizzate nella cache per essere riproposte grazie all'uso dei pulsanti di navigazione.
Con le Single Page Application ciò non vale. Tutto è compreso in un'unica pagina, perciò l'uso dei tasti di navigazione comporta l'uscita dall'intera applicazione. Le soluzioni a questo problema sono due: presentare sulla pagina propri pulsanti di navigazione, oppure ricorrere a ingegnosi programmi che modificano il comportamento del browser, in modo tale da associare le informazioni via via caricate ai pulsanti di navigazione del browser stesso. Nel caso della banca dati di "Biblioteche oggi" abbiamo scelto la prima soluzione perché più semplice; la seconda richiede l'impiego di iframe nascosti, tecnica che abbiamo preferito evitare. La speranza è che la diffusione delle tecnologie Ajax porti a un'evoluzione dei browser tale da ovviare a questi problemi.
Un punto di debolezza delle applicazioni Ajax riguarda l'impossibilità di consentire riferimenti alle singole informazioni attraverso una URI. L'applicazione appare come un oggetto informativo unitario, la cui URI corrisponde alla URL della pagina di avvio.
Questa monoliticità dell'applicazione ha un effetto importante anche per quanto riguarda l'interazione con i motori di ricerca. Le informazioni sulle singole segnalazioni bibliografiche sono "sepolte" nel *deep Web* e non saranno mai indicizzate dai motori di ricerca.
Una soluzione per questo problema è creare dei percorsi alternativi che, nel caso della banca dati di "Biblioteche oggi", corrispondono a un'applicazione che non usa JavaScript. Quando e se si disporrà di una versione del programma con Ajax pienamente accessibile, la versione senza JavaScript verrà dismessa e si realizzerà un accesso alle singole schede bibliografiche basato sugli indici annuali ad uso quasi esclusivo dei motori di ricerca. Ciò assicurerà che le informazioni bibliografiche contenute nella banca dati siano indicizzate anche da questi fondamentali, benché poco raffinati, strumenti per l'individuazione delle informazioni in Internet. La difficoltà di reperire le informazioni attraverso i motori di ricerca corrisponde a una caratteristica generale delle applicazioni Single Page, ma assume particolare rilievo per i database bibliografici.
Questo ci introduce alla seconda tipologia di problemi, quelli specifici delle applicazioni bibliografiche.
Riguardo alle applicazioni relative a database bibliografici e agli OPAC, infatti, gli aspetti critici sono tre:
1) la necessità di ricerca/navigazione a partire da elementi (*authority headings*) dei record trovati;
2) l'ampiezza molto variabile dei risultati delle ricerche;
3) la difficoltà nella scelta della struttura di dati – XML, Json o solo testo – per l'eventuale salvataggio dei singoli articoli.
In un'applicazione bibliografica ogni record recuperato non è mai un punto terminale della ricerca, ma offre molti elementi in grado di attivare ulteriori ricerche, senza che sia possibile in alcun modo definire percorsi privilegiati. L'utente si aspetta di poter attraversare il database in ogni direzione, a partire da qualsiasi notizia. I record bibliografici sono in gran parte composti di elementi sindetici e questa loro caratteristica andrebbe valorizzata. Le applicazioni Ajax devono gestire tali interrogazioni mantenendo la storia di tutte le richieste e le risposte intercorse tra browser e server durante la sessione di ricerca, visto che simile compito non può essere demandato al browser.
La necessità di controllare in modo analitico e costante i dati ricevuti dal server consente la memorizza-

Json, XML e HTML

In sostanza una stringa Json è una stringa di testo strutturata in modo completamente indipendente da ogni linguaggio, ma che usa convenzioni tipiche di una famiglia di linguaggi derivati da C, come C++, C#, Java, JavaScript, Perl Python. I dati sono strutturati in coppie di nome/valore, che nei vari linguaggi sono tradotte come oggetti, record, dizionari, tabelle "hash", liste con chiavi, vettori "hash", oppure in liste di valori, che assumono il ruolo di vettori, liste, sequenze.
La strutturazione dei dati in questo modo può essere facilmente interpretata dai vari linguaggi. In particolare JavaScript la può valutare trasformandola immediatamente in un vettore che può essere percorso dal programma che deve trasformare i dati ricevuti in testo HTML adatto ad essere mostrato in forma tale da divenire informazione utile per l'utente finale.
Un vantaggio rispetto a XML è che la trasformazione può essere fatta percorrendo un oggetto orientato ai dati e non un documento. XML sarebbe risultato più interessante se la trasformazione dei dati avesse potuto essere effettuata in modo economico con XSLT, ma le limitazione che Microsoft Internet Explorer pone all'uso di questo strumento impongono un notevole e ingiustificato aumento dei dati trasmessi. Infatti XSLT può sì consentire di trasformare un documento XML in uno HTML, ma MIE impone che il file di trasformazione sia utilizzato una sola volta, così che dovrebbe essere ricaricato ogni volta che si richiede un elenco o i dati relativi ad un titolo. In questo modo oltre a dover trasmettere ad ogni richiesta file XML più ampi di quelli Json, bisognerebbe ricaricare anche il file XSLT. Il risultato sarebbe spesso un raddoppio dei dati caricati.

Esempi:

a) dati relativi ad un record di articolo in Json

```
{"codicearticolo": "2637", "titolo": "Un \"oggi\" lungo vent'anni" , "resto": "/ Luigi Crocetti" , "fascicolo": "2003 - n. 10", "pagina": " p. 7", "url": "http://www.bibliotecheoggi.it/ 2003/20031000701. pdf", "autori": [["Crocetti Luigi","Crocetti, Luigi"]], "soggetti": [["BIBLIOTECHE OGGI 1983 2003", "Biblioteche oggi - 1983-2003"]], "termini": [["BIBLIOTECHE OGGI","Biblioteche oggi"]]}
```

b) file XML con i dati relativi ad un record di articolo (senza indicazioni di DTD o Schema)

```
<?xml version='1.0' encoding='ISO-8859-1'?><articolo> <codicearticolo>2637</codicearticolo><titoloarticolo>Un "oggi" lungo vent'anni</titoloarticolo><restoarticolo>/ Luigi Crocetti</restoarticolo ><annoarticolo>2003</annoarticolo><fascicoloarticolo>n. 10</ fascicoloarticolo> <paginearticolo>p. 7</paginearticolo><fullarticolo> 200310 00701.pdf</fullarticolo><autori><autore><autord>Crocetti Luigi</autord><autvedi>Crocetti, Luigi</autvedi></autore></autori> <soggetti><soggetto><soggord>BIBLIOTECHE OGGI 1983 2003</ soggord><soggvedi>Biblioteche oggi - 1983-2003</soggvedi></soggetto></soggetti> <termini><termine><termord>BIBLIOTECHE OGGI </termord><termvedi>Biblioteche oggi</termvedi></termine> </termini></articolo>
```

c) file parziale in formato HTML con i dati relativi ad un record di articolo

```
<div id="salva2637"><b>Un "oggi" lungo vent'anni</b> / Luigi Crocetti<br />Fasc.: 2003, n. 10<br />Pagina: p. 7<br /><b><a href= "#"onclick="window.open('http://www. bibliotecheoggi.it/2003/ 20031000701.pdf','titolo','width=600,height=600,resizable=yes,toolbar=yes'); return false"> Testo articolo</a></b><br /><a href="#" onclick= "caricax ('tipo=autore&amp;aut=Crocetti Luigi'); return false"> Crocetti, Luigi</a><br />Soggetto: <a href="#" onclick= "caricax('tipo=ricsog&amp;fcosa=BIBLIOTECHE OGGI 1983 2003'); return false">Biblioteche oggi - 1983-2003 </a><br />Termine: <a href="#" onclick="caricax('tipo= termineunico&amp;ter=BIBLIOTECHE OGGI'); return false"> Biblioteche oggi</a></div><br /><a href="#"onclick ="salva ('salva2637');return false">Salva</a>
```

d) pagina completa in formato HTML con i dati relativi a un record di articolo

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http:// www.w3. org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="it ">
<head>
<title>LIUC Biblioteca Rostoni - Biblioteche oggi</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<meta name="author" content="Biblioteca Rostoni - Università Cattaneo" />
<meta name="DC.Title" content="Indici di Biblioteche oggi" />
<meta name="DC.Creator" content="Biblioteca Rostoni - Università Cattaneo " />
<meta name="DC.Contributor" content="Biblioteche oggi " />
<meta name="DC.Date.Created" content="2007-02-01" />
<meta name="DC.Language" scheme="RFC1766" content ="it" />
<meta name="Keywords" content="periodici, banche dati, biblioteche, biblioteconomia, archivi, documentazione" />
<meta name="Description" content="Sito della Biblioteca Mario Rostoni dell'Universita' Carlo Cattaneo. Banca dati ospitata del periodico Biblioteche Oggi" /><meta name= "DC.Publisher" content="Biblioteca Rostoni - Università Carlo Cattaneo - LIUC" />
<meta name="DC.Type" scheme="DCMIType" content= "text" />
<meta name="DC.Format" scheme="IMT" content= "text/ html" />
<meta name="DC.Identifier" scheme="URI" content= "http: //www.biblio.liuc.it/pagineita.asp?codice=1" />
<meta name="DC.Relation.isPartOf" content="http://www.biblio.liuc.it/" />
<meta name="DC.Rights" content="Tutti i diritti riservati a: Università Carlo Cattaneo – LIUC e Editrice Bibliografica" />
<style type="text/css" title="currentStyle">/*<![CDATA[*/ @import "biboggi.css";/*]]>*/</style>
</head>
<body>
<div id="menu-oggi">
<div><a href="http://www.bibliotecheoggi.it"><img src="/images/testata_bo65h.gif" alt="Mensile di informazione e dibattito" height ="65" width="229" /></a></div>
<div id="top1"><b>Ricerca negli indici di Biblioteche oggi <br /> Torna all'<a href="default.asp">home page</a></b> </div>
<div id="top2"><b>Pagine a cura della<br /><a style="color :#fff" href="http://www.biblio.liuc.it/">Biblioteca 'Mario Rostoni'<br /> dell'Universit&agrave;<br />Carlo Cattaneo - LIUC</a></b></div>
<div><img src="/Images/logobanner.jpg" width="66" height="65" alt="logoLIUC" /></div>
</div>
<div id="corpo">
<h4>Ricerca degli articoli pubblicati su Biblioteche oggi</h4>
<br /><b>Un "oggi" lungo vent'anni</b> / Luigi Crocetti< br />Fasc.: 2003, n. 10<br />Pagina: p. 7<br /><b><a href= 'http://www.bibliotecheoggi.it/2003/20031000701.pdf'> Testo dell'articolo</a></b><br /><a href='ricerche.asp?tipo=autore&aut=Crocetti Luigi'>Crocetti, Luigi</a><br /> Soggetto: <a href='ricerche.asp?tipo=ricsog&fcosa= BIBLIOTECHE OGGI 1983 2003'>Biblioteche oggi - 1983-2003</ a><br />Termine: <a href='ricerche.asp?tipo=termini&ter= BIBLIOTECHE OGGI'>Biblioteche oggi</a><br />
<p>A cura di: Piero Cavaleri e Laura Ballestra<br />
Copyright © Editrice Bibliografica per i dati relativi agli indici e per i testi degli articoli di Biblioteche oggi<br />
Copyright © Università Carlo Cattaneo - LIUC per il software e la realizzazione delle pagine Web</div>
</body></html>
```

zione nel browser degli elementi di qualsiasi file di autorità individuati durante la navigazione per poterli combinare successivamente per ulteriori ricerche. Le dimensione del database di "Biblioteche oggi" sono troppo limitate per rendere conveniente questa strada, ma in altri casi tale possibilità potrà essere utilmente sfruttata.
Un aspetto critico delle applicazioni bibliografiche riguarda l'estrema variabilità dei risultati ottenuti da un'interrogazione. Chi realizza questi programmi deve tener conto che un'interrogazione può produrre liste che vanno da pochissimi elementi a migliaia di elementi. Le prestazioni possono essere influenzate in modo esponenziale dalle dimensioni dei dati da trattare, per cui bisogna valutare attentamente le conseguenze di ogni scelta nei casi estremi di liste molto brevi, che non vanno eccessivamente frazionate, o di liste molto lunghe che devono poter essere rese disponibili in tempi ragionevoli.
Le considerazioni attorno al formato dei dati sono state accennate in varie parti della trattazione, ma è necessario aggiungere alcune considerazioni sugli standard per la formattazione di dati bibliografici. XML-Marc è sicuramente lo standard più importante in questo campo, ma il suo impiego deve essere attentamente valutato rispetto alle necessità di un database di articoli periodici. In ogni caso nessuno standard proposto fa uso del formato Json.
Un problema molto importante, specie nel caso di ricerche bibliografiche che possono estendersi per decine di interrogazioni, è quello della gestione della memoria. Infatti, come già abbiamo detto, nelle Single Page Application l'onere di salvare le informazioni di una sessione risiede in capo al programma. Questo richiede che tutto quello che si ritiene l'utente possa voler rivedere deve essere memorizzato. L'accumulo di tutti questi dati in strutture di JavaScript, array e oggetti, può creare problemi al computer. Il programma deve essere costruito in modo tale da evitare che vengano memorizzati troppi dati e che quelli non più necessari rimangano in memoria.
Nel nostro caso abbiamo ritenuto utile conservare i dati relativi alle ultime cinquanta interrogazioni, lasciando la possibilità di rieseguire quelle precedenti. Questa scelta consente di mantenere per l'utente la possibilità di rivedere tutto ciò che ha cercato, ponendo però un limite alle dimensioni di quello che viene mantenuto in memoria. Inoltre i dati in Json degli elenchi di titoli ormai completamente convertiti in HTML vengo cancellati e gli array che li contengono vengono compattati.
Tutte le interrogazioni svolte vengono conservate per poter essere rieseguite. La disponibilità di questi dati consente, qualora lo si riterrà utile, di creare una funzione di salvataggio di una sessione all'interno di un cookie, le cui dimensioni massime di 4.000 caratteri consentono l'archiviazione di almeno 60/70 query.

Conclusione

In questo articolo e, soprattutto, con la realizzazione della nuova interfaccia della banca dati di "Biblioteche oggi" pensiamo di aver dimostrato l'utilità delle tecnologie Ajax per le applicazioni bibliografiche.
Queste tecnologie possono essere ulteriormente esplorate per realizzare applicazioni bibliografiche più efficienti nel mettere a disposizione degli utenti funzioni che consentano di assumere un ruolo attivo durante il loro utilizzo. Prendere appunti personali o condivisi, recensire documenti, includere una segnalazione bibliografica in un do-

cumento mentre si utilizza un database bibliografico sono operazioni che obbligano gli utenti a cambiare continuamente pagina e il server a gestire una grande quantità di dati per far tornare l'utente al punto di partenza. Ajax consente di fare ciò con grande facilità e assicurando all'utente un'esperienza di estrema semplicità.
La semplicità e la naturalezza con cui si usano le applicazioni Ajax sono il loro punto di forza. Il risultato migliore che chi realizza queste applicazioni può ottenere è quello di constatare che gli utenti non si rendono conto di quanto diverso sia il programma che stanno usando. Più le tecnologie Ajax sono usate con efficacia, più sono trasparenti agli utenti. Quello che speriamo è che nessuno si accorga di quanto la banca dati di "Biblioteche oggi" sia diversa da un normale sito di ricerca bibliografica se non quando si rende conto di come l'esperienza sia stata soddisfacente in termini di facilità e velocità.

Bibliografia

Jason A. Clark, *Building an Ajax application from scratch*, "Computers in Libraries", 26 (006), 10, p. 16-23.
Id., *B102-New Web site tools & technologies: Ajax (Asynchronous JavaScript and XML)*, "Computers in Libraries", 26 (2006), Supp., p. 15-16.
Elizabeth Connor, *Medical librarian 2.0*, "Medical Reference Services Quarterly", 26 (2006), 1, p. 1-15.
Dave Crane – Eric Pascarello, *Ajax in action*, with Darren James, Greenwich, CT, Manning, 2005.
Casey Durfee, *AHAH: when good is better than best*, paper presentato al Seminario "Code4Lib 2006", Corvallis OR, 15-17 feb. 2006, <http://www.code4lib.org/2006/durfee>.
Kevin Curran – Michelle Murray – Martin Christian, *Taking the information to the public through Library 2.0.*, "Library Hi Tech", 25 (2007), 2, p. 288-297.
Jesse James Garret, *Ajax: a new approach to web applications*, "Adaptive Path", 18 febbraio 2005, <http://www.adaptivepath.com/publications/essays/archives/000385.php>.
Fabio Metitieri – Riccardo Ridi, *Biblioteche in rete, istruzioni per l'uso*, Roma – Bari, Laterza, 2003 (nuova ed. 2005).
Id., *Ricerche bibliografiche in Internet, strumenti e strategie di ricerca, OPAC e biblioteche virtuali*, Milano, Apogeo, 1998.
Bruce W. Perry, *Ajax: trucchi e segreti*, Milano, Tecniche Nuove, 2006.
Prototype JavaScript Framework, version 1.5.1, <http://www.prototypejs.org/>.
Diane Vizine-Goetz, *DeweyBrowser*, "Cataloging & Classification Quarterly", 42 (2006), 3/4, p. 213-220.
Jessamyn West, *Ajax: not just another acronym or is it?*, " Searcher", 14 (2006), 1, p. 13-15.
Judith Wusteman – Padraig O'hlceadha, *Using Ajax to empower dynamic searching*, "Information Technology and Libraries", 25 (2006), 2, p. 57-64.

Note

[1] Un modo utile per capire l'evoluzione delle tecnologie degli OPAC è quello di confrontare libri sulle biblioteche pubblicati dieci anni fa con quelli degli ultimi anni. Un esempio si può trovare nelle tre edizioni 1998, 2003 e 2005 del libro di Metitieri e Ridi, *Ricerche bibliografiche in Internet*, e poi *Biblioteche in rete*.
[2] Il termine Ajax è stato introdotto da Jesse James Garret nel febbraio 2005, in un articolo pubblicato sul sito dell'azienda da lui creata, Adaptive Path.
[3] Precursori di Ajax possono essere trovati già negli anni Novanta, per esempio: R. Nelson, *HTML refresh language (HTMLR/1.0)*, <http://tools.ietf.org/html/draft-rfced-exp-nelson-00>. Una proposta che conteneva molti aspetti di ciò che sarebbe stato chiamato Ajax si trova in Marcio Galli – Roger Soares, *Inner-browsing: extending web browsing the navigation paradigm, 2003*, <http://devedge-temp.mozilla.org/viewsource/2003/inner-browsing/index_en.html>.
[4] Dave Crane – Eric Pascarello, *Ajax in action*, 2005.
[5] Una chiara esposizione di questo aspetto, con esempi applicati alle biblioteche, si ebbe nel febbraio del 2006 da parte di Casey Durfee (*AHAH: when good is better than best*), al primo seminario "Code4Lib".
[6] In realtà JavaScript potrebbe essere sostituito con un altro linguaggio di *scripting* supportato dai browser. Il predominio di questo linguaggio dipende dalla migliore, anche se non assoluta, compatibilità dei programmi con i browser più diffusi.
[7] Le ultime versioni attualmente rilasciate di Microsoft Explorer, Firefox, Netscape e Opera supportano senza problemi l'impiego di Ajax e la gestione dell'interfaccia con DHTML e CSS. Per quanto riguarda le versioni precedenti Microsoft Explorer 6 e le versioni di Firefox degli ultimi due anni sono compatibili. Rispetto a Opera le versioni precedenti alla 9 presentavano limitazioni all'uso del metodo post con XMLHttpRequest e all'impiego di XSLT. Netscape nelle versioni precedenti alla 8 dà problemi nella gestione dei CSS.
[8] Un interessante esempio di uso di Ajax per integrare fonti di dati diverse si può vedere all'indirizzo <http://ojax.sourceforge.net/>, progetto di Judith Wusteman e Padraig O'hlceadha, sviluppato presso la UCD School of Information and Library Studies, Dublin, Ireland, con la collaborazione di Industry e il finanziamento della Science Foundation Ireland.
[9] Prototype JavaScript Framework, version 1.5.1.

Abstract

Today bibliographic databases and OPACs have multi-page interfaces to present information to users. Now, using Ajax, is possible to design and create single page applications. The user never needs to change page for requiring more or new information. This new model has the following advantages: no needs to download a whole page for updating only few data; lean streams of data; simple storing and saving data chosen by users; dinamic and rich interfaces. "Biblioteche oggi" database has implemented this solution since spring 2007. The application uses javascript on the client side, json, xml, html for data transmission, asp scripts and MySQL on server side.