HTML5 Mobile App: tecniche per il salvataggio dei dati

L'idea di scrivere questo post nasce dalle numerose richieste che ho ricevuto su come comportarsi per il **salvataggio dei dati** di una **HTML5 mobile** app, le quali presentavano diversi dubbi e perplessità.

Tutte le applicazioni mobile hanno in qualche modo a che fare la manipolazione dei dati: in un videogame abbiamo bisogno di salvare informazioni come punteggi e livelli raggiunti, in un'app per un catalogo prodotti devono essere salvati i dati di ogni prodotto, ecc: insomma, nella maggior parte delle applicazioni abbiamo bisogno di salvare le impostazioni base dell'utente, e così via per una lista infinita di possibili casi.

Il fatto è che i **dati** stanno alla **base** di qualsiasi **web app** (come anche di qualsiasi sito web), quindi vanno in qualche modo gestiti.

Vediamo quali sono le possibili tecniche da utilizzare per il **salvataggio** dei **dati.**

Salvataggio In-memory

Non si tratta di un **salvataggio effettivo,** in quanto le **informazioni** sono assegnate **all'interno** di una **variabile** e vengono perse ogni volta che la **pagina** è **ricaricata** (in termini di una mobile app, ovvero un'applicazione a pagina singola, ogni volta che essa è riavviata).

Questo metodo risulta **utile** per dei **dati statici,** aggiornabili quindi solo al rilascio di una nuova versione dell'app, per i quali solitamente l'interfaccia non permette la modifica da parte dell'utente, a meno chè il fatto che tali modifiche vadano perse al prossimo riavvio sia previsto dallo sviluppatore.

Ecco l'esempio di un modulo **Javascript** che salva in una variabile interna ***(employees)*** dei dati statici ***in-memory*** relativi a dei dipendenti, rendendone possibile il reperimento per **id** e la ricerca

per nome e cognome tramite dei metodi pubblici *(findById* e *findByName):*

Salvataggio tramite localStorage (o web storage)

È una funzionalità **HTML5** che permette di salvare i dati in modo persistente nel browser.

Ma come funziona esattamente?

Semplicemente, è un modo per qualunque **pagina web** (e quindi qualunque **HTML Mobile App** a **Pagina singola)** di **salvare** dei **dati** in **locale** sotto forma di **informazioni** in **coppie** di **chiavi** e **valori.** Come i cookie, tali informazioni persistono anche se l'utente naviga su altre pagine web o chiude e riapre il browser (nel nostro caso anche quando l'utente riavvia l'app o ne utilizza altre). A differenza dei cookie però, i dati non vengono trasmessi al server, ma risiedono solamente sul ***client*** dell'utente (se non si decida naturalmente di inviarli al server in modo manuale).

Il ***localStorage*** è presente nativamente nei browser che lo supportano, quindi per utilizzarlo non abbiamo bisogno di includere plugin esterni o componenti di terze parti, anche se è possibile servirsi di librerie che ne facilitano l'utilizzo.

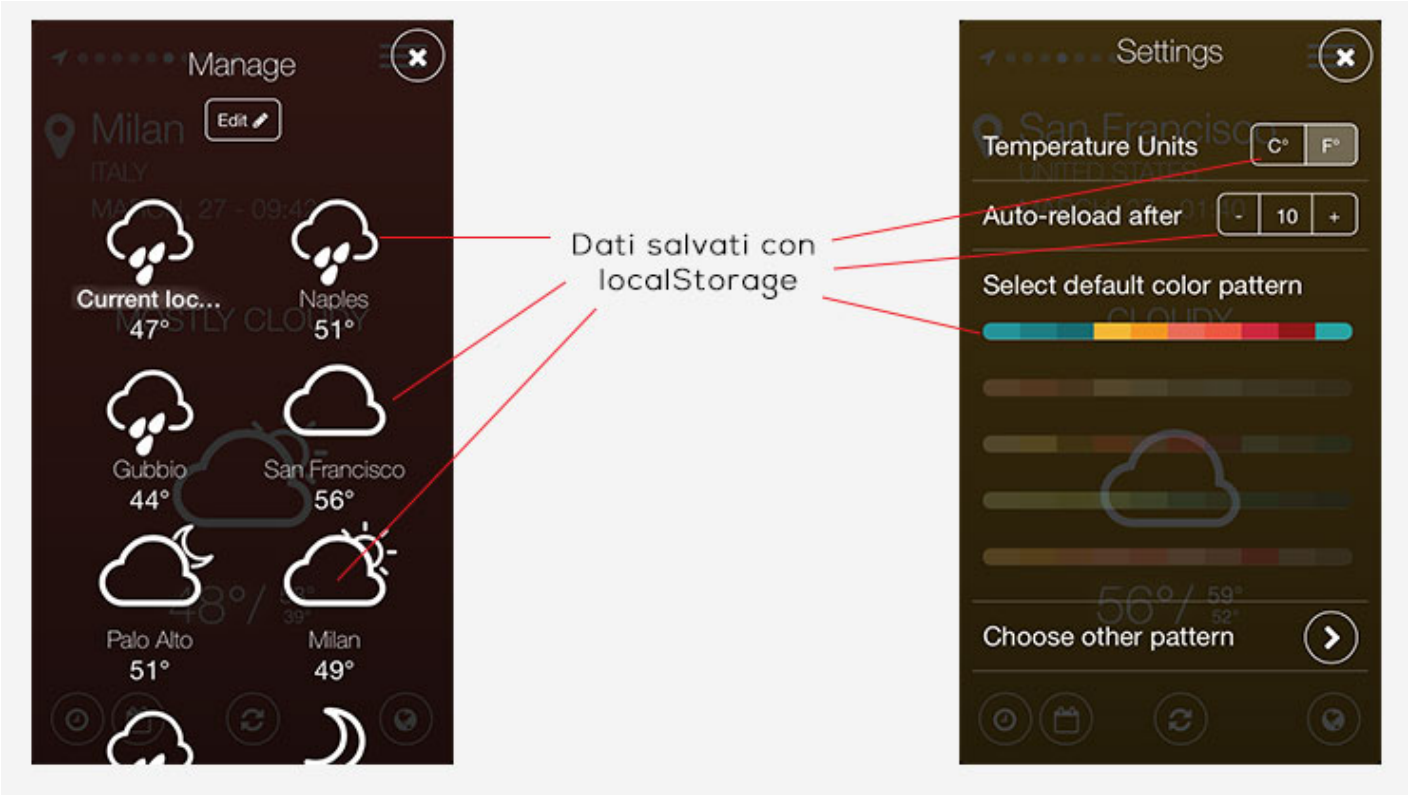
La maggior parte dei browser mobile supportano il ***localStorage.***

Ecco l'esempio del modulo dei dipendenti adattato per l'utilizzo del localStorage, in cui tutto ***l'array*** viene salvato come stringa in corrispondenza della chiave ***"employees":***

Come puoi vedere dal codice, abbiamo inserito il metodo pubblico ***"addItem",*** che permette di aggiungere un nuovo dipendente alla lista, il quale rimarrà disponibile insieme agli altri anche al riavvio dell'applicazione.

Il ***localStorage*** è il metodo che ho usato per il salvataggio delle località e delle impostazioni nella

mia applicazione mobile Tint Weather e per il salvataggio dei dati inerenti le informazioni fisiche dell'utente in Dieta Si o No?, pubblicate entrambe su App Store (puoi saperne di più in questa pagina).



Se anche tu vuoi realizzare e distribuire la tua applicazione, forse ti potrebbe interessare l'e-book che sto scrivendo: HTML Mobile Accelerato.

Salvataggio con un Database HTML

Come probabilmente saprai, è possibile usare due tipi diversi di database lato client in HTML:

1. ***Web Sql:*** mette a disposizione un modello di **API** che permettono di eseguire delle query tramite una variante di linguaggio **SQL.**

2. ***IndexedDB:*** definisce un modello **API** con le quali è possibile lavorare in un database di record con valori semplici ed oggetti gerarchici.

In entrambi i casi i dati vengono salvati esclusivamente **lato client,** sul dispositivo (desktop o mobile) dell'utente.

Purtroppo in questo tipo di salvataggio ci troviamo in una terra ancora instabile:

1. da un lato il ***WebSql*** non sarà più mantenuto in modo attivo dalle ***spec*** dello staff W3C, il che vuol dire che in futuro sarà abbandonato; ecco il link alla pagina ufficiale w3c.

2. dall'altro ***l'IndexedDB*** non è ancora supportato da Mobile Safari per iOS, che invece supporta il WebSql (puoi consultare la tabella delle compatibilità di IndexedDB a questo link); ecco il link alla pagina ufficiale w3c.

Siccome stiamo parlando di **Mobile** App, e siccome il numero di dispositivi Apple rappresenta una grandissima fetta del mercato, è ancora opportuno servirsi del ***WebSql*** per salvare i nostri dati, dirigendosi poi verso l'utilizzo ***dell'IndexedDB*** nel momento in cui esso sarà supportato da tutti i dispositivi.

Anche le versioni desktop di Chrome, Safari e Opera supportano il WebSql, il che ci dà la possibilità di testare eventualmente il salvataggio dati anche su tali tipi di browser, evitando di effettuare ogni volta il **deploy** nel dispositivo.

Vediamo l'esempio del modulo javascript dei dipendenti adattato per il WebSql:

Salvataggio su server via Ajax

I metodi di salvataggio fin qui esposti sono molto utili per salvare dei tipi di dati che non devono essere accessibili da nessun'altro a parte l'utente stesso, e che non possono essere consultati da più di un dispositivo, ma differiscono da un dispositivo all'altro.

Tuttavia, nella maggior parte dei casi lo sviluppatore (o chi distribuisce il prodotto in questione) ha bisogno di **accedere** ai **dati salvati dall'utente,** renderli disponibili ad altri e fruibili anche su diversi dispositivi (magari attraverso un riconoscimento per account) ecc.

Molte volte abbiamo bisogno di **salvare** oppure **ottenere** i **dati** di una **mobile App** da un **server remoto;** che sia attraverso un servizio API creato ad-hoc per la nostra app o un servizio standard di terze parti non fa alcuna differenza, il procedimento è sempre quello di affidarsi a chiamate al server, ovvero alle famose **chiamate Ajax.**

Qualsiasi ***Single-Page-Application*** (e quindi ogni **HTML5 Mobile App)** comunica in remoto esclusivamente con chiamate **Ajax,** per due motivi principali:

1. la pagina non deve mai essere ricaricata e quindi il **DOM** va modificato e manipolato a seconda delle esigenze, invece che richiamare diverse pagine come in un normale sito web

2. si tratta di applicazioni che funzionano interamente grazie a codice ***lato-client,*** e quindi non possono usare i classici linguaggi **lato server** come php, asp, asp.net, C# ecc. per generare codice e dinamico e parti dell'interfaccia.

In un precedente articolo ho parlato di come gestire la connessione di rete per le chiamate Ajax nel

caso di una HTML5 Mobile App:

1. HTML5 Mobile App: come gestire la connessione di rete

In questo **tipo** di **salvataggio** (o reperimento dei dati) il "il ***lavoro sporco"*** viene effettuato sul server (attraverso linguaggi come php, asp, asp.net, C# ecc.), per cui lo sviluppatore dell'applicazione deve solamente preoccuparsi di **inviare** i **dati** da **salvare** ad un url remoto ed attendere eventualmente la risposta di avvenuto inserimento (ovvero la **callback);** per la richiesta dati è ancora più semplice: basterà richiamare solamente l'url (con eventuali parametri aggiuntivi del caso) ed attendere la risposta con i risultati.

Ovviamente è inutile sottolineare che il più delle volte, soprattutto nel caso di sviluppatori autonomi, entrambe le parti client e server sono gestite dalla stessa persona.

Vediamo l'esempio del **modulo javascript** dei dipendenti adattato ad un tipo di salvataggio in remoto:

Conclusioni

Come puoi vedere esistono diversi modi per salvare i dati di un'applicazione e poterne usufruire in qualunque momento.

Ma quali utilizzare?

Ciò dipende dalle tue esigenze:

- se ti basta rendere fruibili dei **dati statici** che non vengono alterati dall'utente, o che comunque non necessitano di mantenere eventuali modifiche al prossimo riavvio, allora non ti basterà il salvataggio **in-memory**
- se il tuo intento è solamente quello di **salvare** delle semplici **preferenze** dell'utente per un solo dispositivo o tipi di dati non troppo complessi, allora potresti prendere in considerazione il **localStorage**

- se invece hai bisogno di una **struttura** più **complessa** di **dati locali,** che preveda una gerarchia, delle relazioni e possa ottenere dei dati incrociati, allora faresti meglio a servirti del **WebSql** o dell'**IndexedDB**

- infine, se vuoi fornire un **servizio cloud** in cui ogni utente può consultare i propri dati da diversi dispositivi, oppure dei dati che subiscono modifiche anche senza il suo intervento, ti affiderai al salvataggio remoto con **chiamate Ajax.**

Naturalmente è possibile (e consigliato) implementare nella propria Mobile App anche più tipi di salvataggio dati.

Potrebbe essere per esempio il caso di un' app che ottiene informazioni remote ma permette di configurare alcuni aspetti tramite un pannello di impostazioni locale: in questo caso dovremmo utilizzare delle chiamate **Ajax** per i **dati remoti** e il **localStorage** o il **WebSql** per salvare i dati delle impostazioni inserire dall'utente.

In altre parole la scelta è a tua discrezione, e dovresti decidere sulla base di due aspetti molto importanti per una buona **user-experience,** soprattutto nel caso di dispositivi mobile: **elevate prestazioni** e **ottimizzazione** dei **dati** passati tra **client** e server (ovvero riduzione dei tempi di attesa).

E tu quali tecniche di salvataggio usi di solito? Scrivilo nei commenti !