Gestire il database attraverso Javascript

Antonio Gallo
info@laboratoriolibero.com

1 Aprire il database

File: 1_creazione_db.html

Per aprire un database basta usare il metodo OpenDatabase:

```
var db = window.openDatabase("note", "1.0", "Note", 200000);
```

Richiede 4 argomenti:

- Il nome del database (note(
- la versione del database che decidete di dargli (1.0)
- una descrizione (Note)
- lo spazio che volete allocare per quel database in byte (200000)

2 Mandare una query e creare una tabella

File: 2_creazione_tabella.html

Per poter mandare una query al database bisogna far partire una transaction e poi eseguire il metodo executeSql.

Ecco il codice:

```
db.transaction(function(tx){

        var table_definition = "create table IF NOT EXISTS note ( id_n INTEGER
PRIMARY KEY AUTOINCREMENT,titolo varchar(100) not null, testo text not null)";

        tx.executeSql(table_definition);

});
```

Nel dettaglio, apro la transazione e definisco la funzione che verrà lanciata quando la transazione sarà avvenuta. A tale funzione verrà passato l'oggetto **tx** (variabile della transazione).

```
db.transaction(function(tx){
```

Uso il metodo executeSql per mandare una query al database:

```
tx.executeSql(table_definition);
```

Dove **table_definition** contiene la query (di creazione della tabella)

3 Inserimento

Per inserire i dati basta usare il metodo executeSql :

Esempio:

```
tx.executeSql("INSERT INTO note (titolo,testo) VALUES
('prima_nona','testo_prima_nota')");
```

Il metodo executeSql accetta 4 parametri:

- La query da eseguire
- (opzionale) Un array di valori da passare alla query e posizionare al posto dei ?
- (opzionale) Una funzione da esegurie se la query è andata a buone fine
- (opzionale) Una funzione da eseguire se la query non è andata a buon fine

Esempio con array di valori:

```
//usando un array di dati
var titolo = "quarta nota";
var testo = "testo quarta nota"
tx.executeSql("INSERT INTO note (titolo,testo) VALUES (?,?)",[titolo,testo]);
```

In questo caso definisco due variabili, **titolo** e **testo**, e nella query predispongo lo spazio per quei valori con due **?**. Poi passo come **secondo argomento** un array avente come elementi le due variabili create.

Esempio con funzione di callback quando la query è stata eseguita:

```
//con una funzione di callback
var titolo = "quarta nota";
var testo = "testo quarta nota"
tx.executeSql("INSERT INTO note (titolo,testo) VALUES (?,?)",[titolo,testo],
function(tx, result){

        console.log("fatto");

});
```

In questo caso la funzione viene eseguita all'avvenuto inserimento. Alla funzione vengono passate due variabili, la variabile di transazione (tx) e una seconda variabile (result) che contiene i dati sul risultato della query.

Per ottenre l'id dell'ultima riga inserita basta usare **result.insertId** come mostrato di seguito:

```
//con una funzione di callback
var titolo = "quarta nota";
var testo = "testo quarta nota"
tx.executeSql("INSERT INTO note (titolo,testo) VALUES (?,?)",[titolo,testo],
function(tx, result){

        console.log(result.insertId);

});
```

4 Modifica ed eliminazione

La modifica e l'eliminazione dei dati si eseguono esattamente come l'inserimento, con il metodo executeSql. Basta usare una query **update** per la modifica e una query **delete** per l'eliminazione.

```
tx.executeSql("delete from note where id_n = 2");
```

oppure:

```
tx.executeSql("delete from note where id_n = ?",[2]);
```

E per la modifica:

```
tx.executeSql("update note set titolo = 'nuovo titolo' where id_n=1");
```

oppure:

```
tx.executeSql("update note set titolo = ?, testo= ? where id_n=3",
["TITOLO","TESTO"]);
```

5 Selezione dei dati

La selezione dei dati è più complessa, ecco un esempio di codice:

```
db.transaction(function(tx){

        tx.executeSql("select * from note",[], function(tx, result){

                if (result != null && result.rows != null) {

                        for (var i = 0; i < result.rows.length; i++)
                        {
                                var row = result.rows.item(i);
```

```
                                console.log("id: "+row.id_n + " - titolo: " +
row.titolo + " - testo: " + row.testo);
                            }
                }

        });

});
```

Nello specifico apro la transazione:

```
db.transaction(function(tx){
```

Eseguo la query:

```
 tx.executeSql("select * from note",[], function(tx, result){
```

In questo caso la variabile **result** conterrà anche i dati estratti. In particolare avrà un attributo **rows** di tipo array che conterrà le varie righe estratte:

Posso quindi controllare che **result** non sia nullo e che il suo attributo **rows** non sia nullo:

```
if (result != null && result.rows != null) {
```

Ciclo sui vari elementi dell'array **result.rows**:

```
for (var i = 0; i < result.rows.length; i++)
```

Utilizzo il metodo item per ottenere un letterale oggetto le cui proprietà sono i campi della tabella e i cui valori sono i valori di tali campi della riga i-esima.

```
var row = result.rows.item(i);
```

Scrivo i valori ottenuti:

```
console.log("id: "+row.id_n + " - titolo: " + row.titolo + " - testo: " +
row.testo);
```

6 Eliminare una tabella

Per eliminare una tabella basta usare il metodo executeSql:

```
var table_definition = "drop table note";

tx.executeSql(table_definition);
```