Les bases de données locales en HTML5

DURBEC Xavier
FARGE Clément

30 novembre 2017

Sommaire

- Historique
  - Origine du problème
  - Solution
- Qu’est ce qui existe
  - et comment ca marche
- Présentation des technos :
  - WebSQL /WebStorage / Indexed Database
- Questions
- Énoncé du TP

Historique

Comment faisait-on avant ?

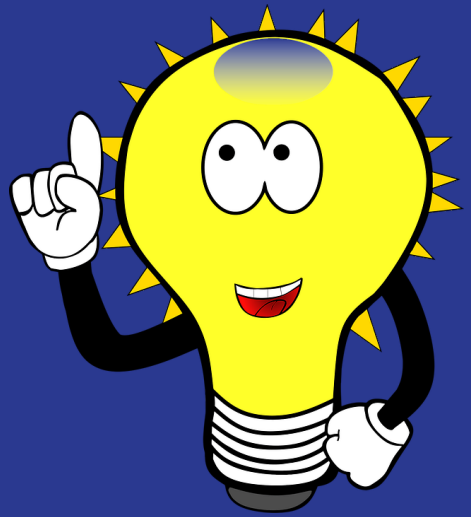
Très bonne question … fichiers (PHP ou javascript) ?

Les cookies (même pas en rêve) ?

Problème résolu

Il existe maintenant plusieurs solutions pour persister des données localement :

- Web storage
- Web SQL
- Indexed database API
- Application cache
- JavaScript ObjectActiveX



WebSQL

Utilise SQLite

Compatible: Chrome, Opera, Safarie et navigateur Android

WebSQL

Déclaration:

```
if(window.openDatabase){
        db = openDatabase(nom_de_la_base_de_données(string),
                version_de_la_base_de_données(string),
                commentaire_de_la_base_de_données(string),
                la_taille_estimé(entier), callback(function)
        );
}
```

l'argument callback() est optionnel.

WebSQL

Changement de Version:

```
db.changeVersion(
        ancien_numero_de_version(entier),
        nouveau_numero_de_version(entier),
        callback_succces(function),
        callback_erreur(function)
);
```

Supporté seulement sur Chrome et Opera.

WebSQL

Transaction :

```
db.transaction(
        callback_de_la_transaction (function),
        callback_erreur_de_la_transaction (function),
        callback_de_succes_de_la_transaction optionnel(function)
);
```

Protège la bdd.

WebSQL

Commande SQL :

```
executeSql(
        requete_sql (string),
        arguments,
        callback_de_succes(function),
        callback_erreur(function)
);
```

WebSQL

Exemple de création d’une table :

```
db.transaction(function(tx) {
        tx.executeSql("CREATE TABLE TableTest (id REAL UNIQUE, text TEXT)", [], function(tx){
                log.innerHTML = '<p>Table1Test created!</p>';
        }, onError);
});
```

WebSQL

Créer une nouvelle entité :

```
tx.executeSql(
        'INSERT INTO foo (id, text) VALUES (?, ?)',
        [id, userValue],
        onSuccess,
        onError
);
```

WebSQL

La fonction SELECT:

```
db.readTransaction(function (t) {
        t.executeSql('SELECT title, author FROM docs WHERE id=?', [id], function (t, data) {
                for(var i = 0; i < data.rows.length; i++){
                        document.getElementById('log').innerHTML = data.rows.item(i).title +
                                '  ' + data.rows.item(i).author;
                }
        });
});
```

WebSQL

La fonction SELECT:

```
data.rows.item(i).title
```

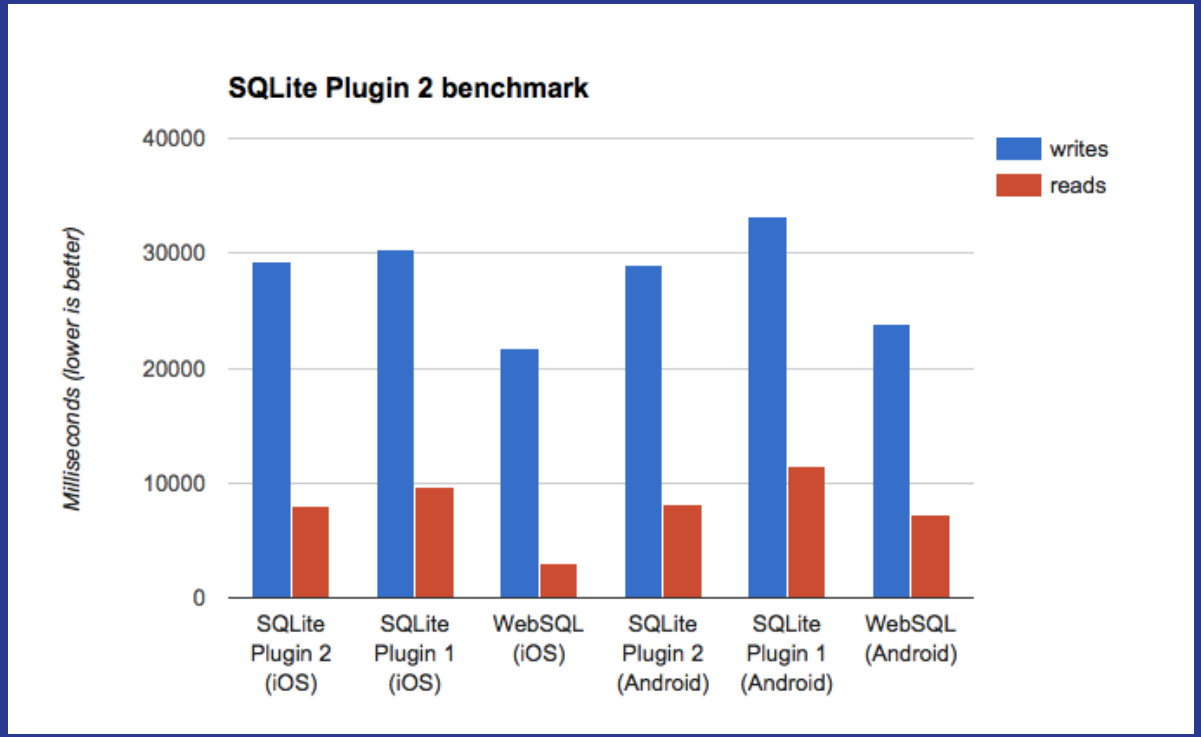
WebSQL est déprécié depuis 2010

“Le consortium W3C a annoncé en novembre 2010 l'arrêt de la spécification. Une alternative possible de stockage standardisée est IndexedDB.”

WebSQL est déprécié, vive le WebSQL!!

- Longtemps la seule manière de faire de la BDDR
- Le Plugin Cordova se base dessus et utilise les même méthodes

WebSQL vs Cordova Plugin SQLite



Web Storage (1)

Déclarer votre BDD :

- Il n’y a rien à faire
- Il suffit de créer une entrée pour créer la BDD
- Une seule BDD par nom de domaine
- La BDD est limitée à 5 Mo par nom de domaine
- Fonctionne sur le principe de clé / valeur

Web Storage (2)

Insérer des données :

```
localStorage.setItem(key, data);
```

La clé et la data sont des chaînes de char !

Web Storage (3)

Récupérer des données :

```
localStorage.getItem(key);
```

Le getItem retourne une chaîne de caractères !

Web Storage (4)

Effacer une donnée :

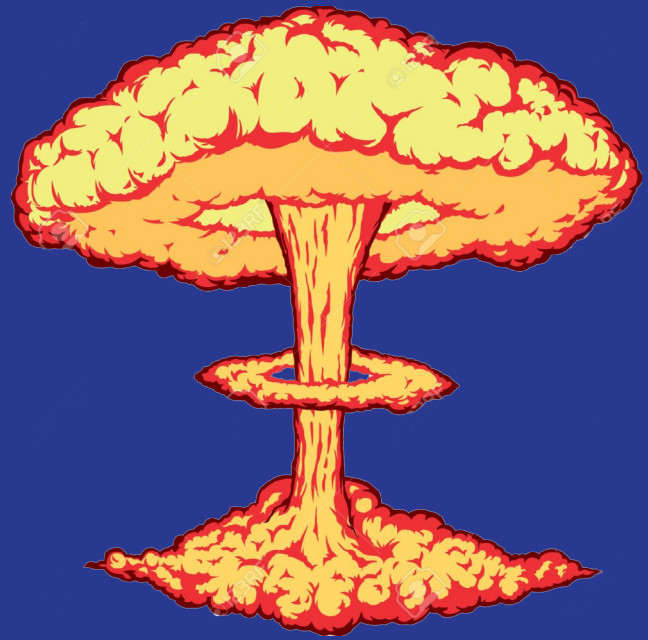
```
localStorage.removeItem(key);
```

Web Storage (5)

Effacer la base :

```
localStorage.clear();
```

Efface TOUTE la base !



Indexed DataBase (1)

Avec quel navigateur on travaille :

```
var indexedDB = window.indexedDB || window.mozIndexedDB ||
    window.webkitIndexedDB || window.msIndexedDB || window.shimIndexedDB;
```

Créer la BDD :

```
var open = indexedDB.open("CarnetAdresses", 1);
```

Indexed DataBase (2)

L’objet open :

```
var open = indexedDB.open("CarnetAdresses", 1);
```

Possède les méthodes :

- onupgradeneeded
- onsuccess
- onerror
- oncomplete

Indexed DataBase (3)

Si la méthode open s’est bien passé :

```
open.onupgradeneeded = function() {
    var db = open.result;
    var store = db.createObjectStore("Entry", {keyPath: "mail"});
};
```

Indexed DataBase (4)

On peut maintenant travailler (put):

```
open.onsuccess = function() {
    // Start a new transaction
    var db = open.result;
    var tx = db.transaction("Entry", "readwrite");
    var store = tx.objectStore("Entry");
    store.put({nom:$("#nom").val(),
        prenom:$("#prenom").val()
    });
}
```

Indexed DataBase (5)

On peut maintenant travailler (get):

```
var getEntry = store.get($("#key").val());
getEntry.onsuccess = function() {
    console.log(getEntry.result.nom);
};
```

Indexed DataBase (6)

On peut supprimer des données :

```
open.onsuccess = function() {
    var db = open.result;
    var transac = db.transaction("Entry", "readwrite");
    var action = transac.objectStore("Entry");
    action.delete($("#key").val());
};
```

Indexed DataBase (7)

Une fois la transaction terminée :

```
tx.oncomplete = function() {
    db.close();
};
```

Indexed DataBase (8)

Pour les plus téméraires :

```
$("#delBDDIDB").click(function () {

    indexedDB.deleteDatabase("CarnetAdresses");
});
```

Avez vous des questions ?

A vous de jouer !