HTML5: Web Storage

By: John Temoty Roca

Web Storage

- The Web SQL Database API isn't actually part of the HTML5 specification but it is a separate specification which introduces a set of APIs to manipulate client-side databases using SQL.
- Web SQL Database will work in latest version of Safari, Chrome and Opera.

The Core Methods:

- There are following three core methods defined in the spec that I.m going to cover in this tutorial:
- openDatabase : This method creates the database object either using existing database or creating new one.
- transaction: This method give us the ability to control a transaction and performing either commit or rollback based on the situation.
- executeSql : This method is used to execute actual SQL query.

Opening Database:

- The *openDatabase* method takes care of opening a database if it already exists, this method will create it if it already does not exist.
- To create and open a database, use the following code:

```
var db = openDatabase('mydb', '1.0', 'Test DB', 2 * 1024 * 1024);
```

- Above method took following five parameters:
  - Database name
  - Version number
  - Text description
  - Size of database
  - Creation callback

Executing queries:

- To execute a query you use the database.transaction() function. This function needs a single argument, which is a function that takes care of actually executing the query as follows:

```
var db = openDatabase('mydb', '1.0', 'Test DB', 2 * 1024 * 1024);

db.transaction(function (tx) {
        tx.executeSql('CREATE TABLE IF NOT EXISTS LOGS (id unique, log)');
});
```

- The above query will create a table called LOGS in 'mydb' database.

INSERT Operation:

- To create entries into the table we add simple SQL query in the above example as follows:

```
var db = openDatabase('mydb', '1.0', 'Test DB', 2 * 1024 * 1024);
db.transaction(function (tx) {
            tx.executeSql('CREATE TABLE IF NOT EXISTS LOGS (id unique, log)');
            tx.executeSql('INSERT INTO LOGS (id, log) VALUES (1, "foobar")');
             tx.executeSql('INSERT INTO LOGS (id, log) VALUES (2, "logmsg")');
});
```

- We can pass dynamic values while creating entering as follows:

INSERT Operation:

```
var db = openDatabase('mydb', '1.0', 'Test DB', 2 * 1024 * 1024);
db.transaction(function (tx) {
        tx.executeSql('CREATE TABLE IF NOT EXISTS LOGS (id unique, log)');
        tx.executeSql('INSERT INTO LOGS (id,log) VALUES (?, ?'), [e_id, e_log];
 });
```

- Here e_id and e_log are external variables, and executeSql maps each item in the array argument to the "?"s.

READ Operation:

- To read already existing records we use a callback to capture the results as follows:

```
var db = openDatabase('mydb', '1.0', 'Test DB', 2 * 1024 * 1024);
db.transaction(function (tx) {
        tx.executeSql('CREATE TABLE IF NOT EXISTS LOGS (id unique, log)');
        tx.executeSql('INSERT INTO LOGS (id, log) VALUES (1, "foobar")');
        tx.executeSql('INSERT INTO LOGS (id, log) VALUES (2, "logmsg")');
});
db.transaction(function (tx) {
        tx.executeSql('SELECT * FROM LOGS', [], function (tx, results)
        {
                var len = results.rows.length, i; msg = "<p>Found rows: " + len + "</p>";
                document.querySelector('#status').innerHTML += msg;
                for (i = 0; i < len; i++){
                        alert(results.rows.item(i).log );
                }
        },
null); });
```