Mobile Web

SEM2220 - Assignment 1

Alexander D Brown (adb9)

Contents

1 Introduction **3**

2 Implementation **3**
2.1 Implementing the Database Query 3
2.2 Problems Encountered 4

3 Preview **4**

4 Testing and Debugging **4**
4.1 Debugging Using the Developer View in Chromium 4
4.2 Testing on Android Emulator 4

5 Evaluation **4**

1 Introduction

This report shows the process undertaken to change the Conference Web Application (produced by Chris Loftus) from a statically generated list to one loaded dynamically using JavaScript from a WebSQL database.

The Conference Web Application is a website which was designed using progressive enhancement to provide content to mobile devices as well as desktop browsers. It uses the jQueryMobile framework to give a native feel for mobile users.

The process involved the implementation of some JavaScript code to query the WebSQL database created in HTML5 storage when the Conference Web Application is initially loaded. The results of this query are then dynamically rendered in the sessions page as a part of a jQueryMobile list view.

2 Implementation

This section describes the process taken to implement the dynamic loading of sessions into the session view from the pre-existing WebSQL database. There were two main steps to do this:

1. Query the database using standard SQL statements.
2. Render the results of the query.

Because a lot of the code was already written this was a simple case of implementing two methods. The method to query the database handled the building of SQL to perform the query and use a callback method to handle the results of this query. This callback method also needed implemented and read the each of the results and appended them to a jQueryMobile list view object using jQuery methods.

2.1 Implementing the Database Query

The `DataContext.js` file handles the data handling for the application, on the first load of the whole site the database is built from hard-coded values in the file. It is also responsible for performing database queries and transactions and therefore is the location which the sessions query is required.

To start with a simple SQL statement was designed, based on the hints from the original author which would select everything from the sessions table where the session was on the first day of the conference.

```
SELECT * FROM sessions WHERE sessions.day_id = '1'
                       ORDER BY sessions.starttime ASC
```

From this it is a simple matter of calling a method from the transaction method as shown:

```
var querySessions = function(tx) {
  var sql = // SQL Statement
  tx.execute_sql(sql, [], callback, error_callback);
};
```

Where the callback is the rendering function.

To make this statement a little more secure, as recommended by the API, it was changed to be:

```
var querySessions = function(tx) {
  var sql = "SELECT * FROM sessions WHERE session.day_id = ?
            ORDER BY sessions.starttime ASC";
  tx.execute_sql(sql, [1], callback, error_callback);
};
```

Though the query is completely isolated from user input, it may not be in the future.

To enhance this further, joining the days table to include the name of the day that the session was on. This could have been hard coded in the rendering function, but again it may not remain limited to the first day so doing this dynamically will be a benefit in the future.

```
var querySessions = function(tx) {
  var sql = "SELECT sessions.* days.day FROM sessions, days
            WHERE sessions.day_id = ?
            AND days._id = sessions.day_id
            ORDER BY sessions.dayid
            AND sessions.starttime ASC";
  tx.execute_sql(sql, [1], callback, error_callback);
};
```

2.2 Problems Encountered

3 Preview

4 Testing and Debugging

4.1 Debugging Using the Developer View in Chromium

4.2 Testing on Android Emulator

5 Evaluation