Creating Offline Web Applications Using HTML5

Sam Marshall

A thesis presented in partial fulfillment of the requirements for the degree of

Master of Information Sciences
in
Computer Science

Institute of Information and Mathematical Sciences
Massey University – Albany
North Shore 102-904, Auckland, New Zealand
Email: sam.marshall@windowslive.com
Tel: +64 9 414 0800 Fax: +64 9 441 8181

March 1, 2013

Abstract

With the proliferation in the use of mobile devices, browser based applications are becoming the ideal information system for both individuals and organization. Web applications are platform independent and easy to deploy so can be accessed from any device that has a browser. A large number of businesses are now offering cloud services to deliver their software, which are on-demand and pay-as-you-go solutions. The increase in this trend is adding a huge economic and bandwidth challenge to both the network provider and consumer.

While traditional web applications work when they are online, it is however essential for these applications to be available both online and offline modes. With this explosion in the use of mobile devices, the ability of these applications to work offline is especially important in situations where there is intermittent or no network availability.

In this thesis we discuss ways of developing offline web applications. We also propose a method of implementing a wrapper that simplifies the currently proposed W3C's HTML5 client-side database API, IndexedDB, by providing a fluent interface with a Language Integrated Query (LINQ) feel. In cases where synchronization of the client-side data with the server database is a requirement, conflict resolution becomes a bit challenging. We discuss techniques for synchronizing the data that is stored at the client during offline mode with the server database.

Keywords: Web applications, HTML5, Client side storage, Browser databases, IndexedDB, Database replication/synchronization

Acknowledgements

I would like to express my thanks to some people who supported me. First, I thank Prof. Ken Hawick for supervisory, advisory support and for giving me the opportunity to write about this interesting topic.

Thanks also to my wife, who has been there for me and our two beautiful kids, Sean and Chantelle-Rose despite not seeing much of me.

And last but not least I would like to thank my parents and my friends for their ongoing encouragement, and their willingness to understand my research topic, even though they do not have any clue of what I am talking about.

Contents

1 Introduction **8**

1.1 Overview 8

1.2 Problem Definition 10

1.3 Objectives 12

1.4 Security And Privacy 14

1.5 Volatility of User data 15

1.6 Thesis structure 16

2 Literature Review **17**

2.1 Introduction 17

2.1.1 Web Application 17

2.1.2 Online Web Application 20

2.1.3 Offline Web Application 21

2.1.4 Latency 25

2.1.5 Web Caching 25

2.1.6 Web Application Performance 26

2.1.7 Web Application Scalability 26

2.1.8 Web Performance Rules 27

2.2 Technology Drivers 27

2.2.1 Cloud Computing: Software as a Service (SaaS) 27

2.2.2 Mobile Computing 27

2.3 Database Synchronization 28

2.3.1 Introduction . . . 28
2.3.2 Data integrity . . . 30
2.3.3 Database synchronization techniques . . . 30
2.4 Current Issues . . . 31
2.4.1 Web architecture challenges . . . 31
2.4.2 Functional dependencies . . . 32
2.4.3 Data dependencies . . . 32
2.5 Existing Solution Options . . . 32
2.5.1 Dojo Offline Toolkit . . . 32
2.5.2 Google Gears . . . 33
2.5.3 Web SQL Database . . . 34
2.6 HTML5 . . . 35
2.6.1 Introduction . . . 35
2.6.2 Hardware Acceleration . . . 36
2.6.3 Local File Access . . . 37
2.6.4 IndexedDB . . . 38
2.6.5 Cookies . . . 39
2.6.6 Web Storage . . . 40
2.6.7 Application Cache . . . 42
2.6.8 Web Workers . . . 43
2.6.9 Long polling . . . 44
2.6.10 Web Sockets . . . 44
2.6.11 XMLHttpRequest 2 . . . 47

3 Methodology **49**
3.1 Introduction . . . 49
3.1.1 Important Design Aspects . . . 51
3.2 Designing for Performance and UI Responsiveness . . . 51
3.3 Javascript Libraries . . . 56
3.3.1 jQuery . . . 56
3.3.2 Knockout.js . . . 56

3.3.3 Model View ViewModel Design Pattern . . . 56
3.4 HTML5 . . . 58
3.5 IndexedDB . . . 59
3.5.1 ACID Properties of Transactions . . . 60
3.5.2 Asynchronous . . . 61
3.5.3 Synchronous . . . 62
3.6 Application Cache . . . 62
3.6.1 Problems with App Cache . . . 64
3.6.2 Improving App Cache . . . 65
3.6.3 Leveraging Browser Cache . . . 66
3.7 Designing a Fluent Interface . . . 69
3.7.1 Domain Specific Language (DSL) . . . 70
3.7.2 Method Chaining . . . 72
3.7.3 Language Integrated Query (LINQ) . . . 73
3.7.4 Lazy Load Pattern . . . 74
3.8 Data persistence . . . 74
3.8.1 Integrity and Reliability . . . 75
3.9 Database Synchronisation . . . 75
3.9.1 Data Consistency and Application Responsiveness . . . 76
3.9.2 Client-side Transaction Logging . . . 76
3.9.3 Receiving Server Updates . . . 76
3.9.4 Change Tracking . . . 77
3.9.5 Conflict Detection and Resolution . . . 78
3.9.6 Prioritizing Data Exchange . . . 80
3.9.7 Background Synchronization . . . 80
3.9.8 Security . . . 80
3.9.9 Error Handling and Recovery . . . 81

4 Privacy and Security **82**

4.1 Privacy . . . 82
4.1.1 Introduction . . . 82

4.1.2 Privacy By Design 82
4.1.3 User Centred Design 83
4.1.4 User information confidentiality 83
4.1.5 Control and log access 83
4.1.6 Sensitivity of data 84
4.2 Security 84
4.2.1 Introduction 84
4.2.2 Injection Attacks 85
4.2.3 File System API Security Considerations 91
4.2.4 IndexedDB Security 92
4.2.5 Web Sockets API 92
4.2.6 Countermeasures 92

5 Discussions And Conclusions **94**
5.1 Discussion 94
5.1.1 Design Considerations 95
5.1.2 Implementation risks 96
5.1.3 Browser Differences 96
5.1.4 Performance and Scalability 97
5.1.5 Security 97
5.1.6 Client-side Development and Debugging 97
5.2 Summary And Conclusions 98

List of Figures

1.1 Page Not Found 11

2.1 HTML DOM Tree (23) 19
2.2 Mobile Users versus Desktop Users (28) 20
2.3 Core elements of an online web application 21
2.4 Core elements of an offline web application 24
2.5 IndexedDB Browser Compatibility(37) 40
2.6 Typical WebSocket handshake: web client request and web server response (50) 45
2.7 Network Throughput: Polling vs WebSocket (78) 46
2.8 Latency: Polling vs WebSocket(78) 47

3.1 Extended: Core elements of an offline web application 52

Listings

2.1 Creating a table with Web SQL 34
2.2 Retrieving data with SQL select 34
2.3 Storing a JSON Object 42
3.1 How to check online status 50
3.2 Subscribing to window event 50
3.3 Example of a Manifest File 63
3.4 Traditional Way of Querying IndexedDB 69
3.5 Proposed Way of Querying IndexedDB 70
3.6 Method Chaining Example 72