

HELSINKI UNIVERSITY OF TECHNOLOGY
Department of Computer Science and Engineering
Telecommunications Software and Multimedia Laboratory

Samuli Karevaara

Implementing asynchronous scripting techniques in classic web applications

Master's Thesis

Hollola, June 29, 2006

Supervisor: Professor N.N.

Instructor: M.Sc. N.N.

HELSINKI UNIVERSITY OF TECHNOLOGY Department of Computer Science and Engineering		ABSTRACT OF MASTER'S THESIS	
Author Samuli Karevaara		Date	
		Pages	
Title of thesis Implementing asynchronous scripting techniques in classic web applications			
Professorship T-111		Professorship Code ?	
Supervisor Professor N.N.			
Instructor M.Sc. N.N.			
<p>In the late 1990s there was a lot of talk about a network computer. All of the applications would be installed on a server computer, and the computer at home would only use these remotely installed programs. This didn't happen.</p> <p>However, from the beginning of the World Wide Web there has been more and more applications that can be used with a web browser. Recent advances in the networking and browser technology have given an opportunity to build applications that move from the classic page based web application model to a more asynchronous model of data processing.</p> <p>The asynchronous techniques require a lot more from the browser than the classic page based web application model. The page based model is also simpler to implement as it has all of it's business logic on the server side only. Thus the classic model will prevail for quite a while.</p> <p>This thesis identifies the situations where the asynchronous techniques can be integrated with the classic web applications. The thesis will also outline what kind of user interface changes are required, how and if they change the usability and availability of the application and do they have an effect on the application efficiency.</p> <p>Along with the user interface analysis a software architecture analysis will be made. What fundamental changes, if any, should be made to the software architecture. A study of the suitability of the model-view-controller design pattern is carried out.</p> <p>The possible benefits are two-fold: increased efficiency for the users and increased efficiency for the server. The unavoidable drawback is that some of the user interface elements and the code structures will have to be duplicated for the server side and for the client side.</p> <p>This thesis will use a widely used open source course management system as the basis of the user interface and software architecture analysis. Necessary changes will be made to the course management system to allow the use of asynchronous model. Both the user and server efficiency changes will be measured.</p> <p>While the thesis mostly concentrates on one classic open source web application, it represents a very common base of classic applications, and the results of the analysis can be applied to a range of other web applications also.</p>			
Keywords Web applications, asynchronous scripting			

HELSINGIN TEKNILLINEN KORKEAKOULU Tietotekniikan osasto	DIPLOMITYÖN TIIVISTELMÄ	
Tekijä Samuli Karevaara	Päivämäärä	
	Sivuja	
Diplomityön otsikko Asynkronisten skriptaustekniikoiden toteuttaminen klassisissa web-sovelluksissa		
Professuuri T-111		
Valvoja Professori N.N.		
Ohjaaja DI N.N.		
...		
Avainsanat		

Table of Contents

1 Introduction.....	1
1.1 Problem and goals.....	1
1.2 Structure of the thesis.....	1
2 Background.....	2
2.1 Internet.....	2
2.2 World Wide Web.....	2
2.3 Classic web applications.....	3
2.4 Software design patterns.....	3
2.5 Usability and related issues.....	3
3 Asynchronous web applications.....	4
3.1 Changing the page contents dynamically.....	4
3.2 Asynchronous data transfer and processing.....	4
3.3 Asynchronous scripting frameworks.....	4
3.4 User interfaces for asynchronous web applications.....	4
3.5 Creating a simple asynchronous web application.....	4
3.6 Examples of popular asynchronous web applications.....	4
4 Asynchronous scripting for classic web applications.....	4
4.1 Motivation.....	5
4.2 Design patterns for web applications.....	5
4.3 User interface and an asynchronous view layer.....	5
4.4 Identifying the target features.....	5
4.5 Showing the application state.....	6
4.6 Software architecture changes.....	6
4.7 Separating the model layer.....	6
4.8 Concurrency and caching issues.....	6
4.9 Procedural versus object oriented model.....	7
5 Analysis of the results.....	7
5.1 User efficiency.....	7
5.2 Server efficiency.....	7
5.3 Increased development efforts.....	7
5.4 Refactoring for web services.....	7
6 Conclusion.....	7
6.1 Recommendations for future studies.....	7

Foreword

...

This thesis is completely done using open source software. The thesis itself is written with OpenOffice.org. The diagrams in this thesis are done with Inkscape. The case study is done on Moodle. Software development is done mostly using the Vim editor, Subversion revision control and TortoiseSVN Subversion client.

...

Lastly and most importantly, I'd like to thank my wife for her support, insightful conversations and advice on research techniques and bibliographical referencing styles.

Abbreviations and acronyms

FTP	File Transfer Protocol
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IP	Internet Protocol
TCP	Transmission Control Protocol
WWW	World Wide Web

Glossary of terms

Hypertext Text that can have links to other texts. These links can be followed by the reader to fetch the linked texts.

1 Introduction

In the early days of computing the computer users only accessed terminals which were connected to a mainframe computer. The actual computing was done on the mainframe computer, and the terminal was just for the program input and output. Terminals were usually connected to the mainframe computer via a local area network.

When the processing power and the price of the personal computers reached comfortable levels in the 80's, computers were installed on people's homes. The program logic was completely moved to the personal computers. The wide area network was very slow, which also attributed to this.

Advances in the network technologies and the advent of the World Wide Web lead many to believe that the networking computing model would return.

Asynchronous web application techniques have proven benefits [lähde?], so they should not be dismissed offhand.

The purpose of this thesis is to identify the situations where implementing asynchronous scripting techniques in classic web applications have more benefits than drawbacks, and to analyze the type of changes needed in the application user interface and software architecture.

While this thesis will concentrate on a widely used open source course management system will be used as basis for the work, the results can be applied to a wide range of web applications.

The possible benefits are two-fold: increased efficiency for the users and increased efficiency for the server.

1.1 Problem and goals

How can asynchronous techniques be used effectively in classic web applications without duplicating the development effort?

The solution should

- not break the classic model
- not double the development efforts
- and have measurable benefits.

Because of the fundamental differences between the classic web application model and the asynchronous model they can't be seamlessly combined. Thus an added development effort can't be avoided.

The solution will therefore concentrate on minimizing the added development effort and maximizing the gained benefits.

1.2 Structure of the thesis

Chapters 2 and 3 form a theoretical background for the thesis.

Chapter 2 contains an explanation of the fundamental concepts behind web applications and the web infrastructure.

Chapter 3 outlines the nature of the asynchronous web applications. features that will be

implemented in a classic web application. The chapter also contains examples of partial solutions to the thesis problem, as well as examples of some popular current asynchronous web applications and the solutions behind them.

Chapter 4 explains how the asynchronous techniques mentioned in chapter 3 can be applied to classic web applications also. A model solution against a selected widely used classic web application will be crafted and explained. The required software architectural changes are outlined.

Chapter 5 analyses the benefits and drawbacks of the work done in chapter 4.

Chapter 6 contains a summary conclusion of the work and thoughts on possible future studies.

2 Background

Vannevar Bush (1945) ... over half a century ago .

2.1 Internet

The Internet is a worldwide collection of computer networks that are connected together. The data is transmitted between the networks in data packets using a protocol called the Internet Protocol (IP). The IP data packets form the basic information flow of the Internet. Applications that utilize the Internet usually also use a higher level protocol, like File Transfer Protocol (FTP) or Hypertext Transfer Protocol (HTTP) (Comer 2001, p. 2-7).

Client-Server paradigm

In the client-server paradigm one computer program acts as a client and initiates communication with a server. The server is another computer program that waits for the clients to communicate with it. When the server receives information from a client, it computes the information according to the application logic and sends the result back to the client. (Comer & Stevens 2001, p. 10-11)

Stateless and stateful servers

2.2 World Wide Web

The World Wide Web (WWW or web) in its basic form is a collection of different types of data, like pictures, text, sound and video, that can be read with a web browser. Data in the web is stored on web servers. Pieces of the data in the World Wide Web are generally called web pages.

Each individual web page is given a unique address, called the Uniform Resource Locator (URL). The URL for a web page specifies both the web server that holds the page and the location of the page on the web server. (Berners-Lee 1996)

General architecture

Web servers are connected to each other over the Internet. (Berners-Lee 1996)

HTTP

Web browsers send requests to fetch web pages from web servers using a protocol called the Hypertext Transfer Protocol. (Comer 2001)

Server side programming

Very early on it was realised that the web pages could be dynamically produced at the time when they are served to the web browser. A trivial application of this would be to print today's date and the current time on the web page automatically.

The automatic producing of the current time on a web page is rarely of any use, but a more useful application of programmatically produced web pages could be to fetch phone numbers from a call central database and serve them as web pages. A very early web server software made in 1991 was in fact a web browsable gateway to the CERN phone book. (Berners-Lee 1996)

Client side scripting

2.3 Classic web applications

Implementing stateful servers

Sessions, cookies...

Typical web application architecture

2.4 Software design patterns

Model-View-Controller pattern

Separating the data, the presentation and the application logic.

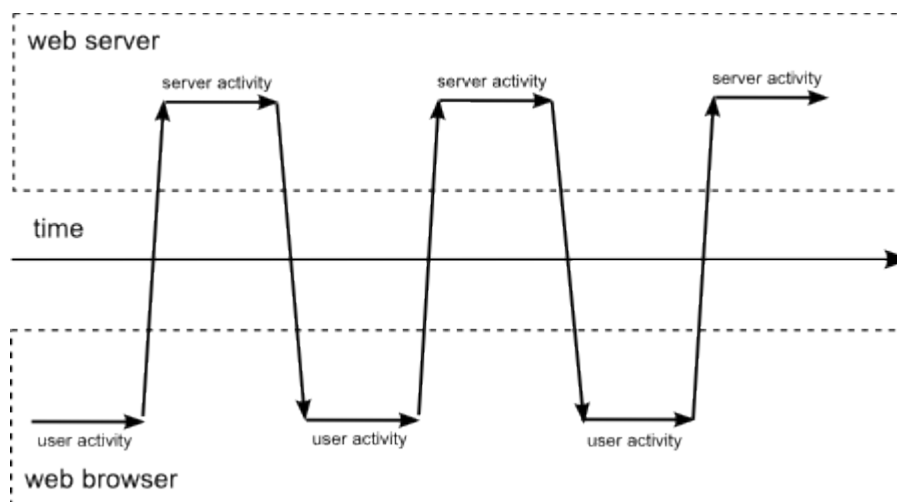
Page-Controller pattern

2.5 Usability and related issues

Usability**Accessibility****Efficiency**

Solutions that are very inefficient on the server will result in very slow applications. Slow applications are generally considered to have a lower usability than quick applications. [lähde?]
[user efficiency versus server efficiency]

3 Asynchronous web applications



3.1 Changing the page contents dynamically

3.2 Asynchronous data transfer and processing

3.3 Asynchronous scripting frameworks

3.4 User interfaces for asynchronous web applications

3.5 Creating a simple asynchronous web application

3.6 Examples of popular asynchronous web applications

4 Asynchronous scripting for classic web applications

4.1 Motivation

4.2 Design patterns for web applications

4.3 User interface and an asynchronous view layer

4.4 Identifying the target features

Deletions from a list

Example: Deleting a course

Small changes in the program state

Example: Marking a discussion thread as read

Moving objects on screen

Example: Moving the course side blocks

Example: Moving the course modules

Form validation

Read-only operations

Example: Browsing calendar events

Pagination of large tables

Example: Browsing the access logs

4.5 Showing the application state

Progress indicators

Error messages

Example: Marking a discussion thread as read

Script generated dialogs

Latency issues

4.6 Software architecture changes

4.7 Separating the model layer

4.8 Concurrency and caching issues

4.9 Procedural versus object oriented model

5 Analysis of the results

5.1 User efficiency

5.2 Server efficiency

5.3 Increased development efforts

5.4 Refactoring for web services

6 Conclusion

6.1 Recommendations for future studies

References

- Berners-Lee, T. (1996), WWW: Past, Present and Future. IEEE Computer 1996, 10. pp. 69-77.
- Bush, V. (1945), As we may think. Atlantic Monthly 176, 1 (July 1945). pp. 101-108.
- Comer, D. (2001), Internetworking with TCP/IP, Volume 1, Principles, protocols, and architectures. Prentice Hall.
- Comer, D. & Stevens D. (2001), Internetworking with TCP/IP, Volume 3, Client-server programming and applications. Prentice Hall.
- Fournier, R. (1998), A methodology for client/server and Web application development. Yourdon Press.
- Fowler, M. (2003), Patterns of Enterprise Application Architecture. Pearson Education, Inc.
- Garrett, J. (2005), Ajax: A New Approach to Web Applications. Web reference. Retrieved July 28, 2006 from <http://adaptivepath.com/publications/essays/archives/000385.php>.