ShaRefWeb
A Web Interface for the ShaRef Service

Master's Thesis by Thierry Bücheler
under supervision of Prof. Dr. Bernhard Plattner
and Dr. Erik Wilde

October 20, 2005

This master's thesis has been written at the Computer Engineering and Networks Laboratory (TIK) at the Federal Institute of Technology (ETH) Zürich, Switzerland.

Contents

0.1 Introduction

While traditional libraries often concentrate on the repository and retrieval facets of their work, for many scientists it is also very important to be able to manage and maintain references to the information they work with, such as bibliographic information and Web bookmarks. ShaRef aims at providing a solution (usable Web-based as well as offline) for creating, managing, and sharing reference information. This information can be exported to other applications (such as word processing software or Web browsers). ShaRef is based on XML technologies for providing an open platform with powerful importing and exporting features. Web-based publishing features allow Web access and the creation of selected reference lists such as reading lists for lectures and seminars or publication lists for specific research areas.

This report was written during a master's thesis at ETH Zürich which involved the development of the Web interface for ShaRef and its documentation and it gives some information on the condition and context of the development process.

Chapter 1

The Project

1.1 Objectives and Procedure

The *Objectives* section states my (and the project's) goals for this master's thesis.

To work efficiently, I followed a structured workflow for the project. I broke down the project in parts to set milestones and monitor the progress. The desciption of this process is described in the section *Procedure*.

1.1.1 Objectives

The goal of the master's thesis was to create a working distribution of a Web interface for the ShaRef software.

Development of the ShaRef software and the Web client was concurrent, so features added to the software could be added to the Web client as soon as they where developed for the main application. This means that the number of features added during the period of the master's thesis was only roughly defined and could change during project time (which it actually did).

The documentation (including this paper) was a crucial part of the work because the application would be handed to another department at ETH after completion of the project at the end of 2005.

1.1.2 Procedure

Understand the project

In a first step the architecture of the ShaRef software and its usage scenarios had to be undestood. Since the development was taking place almost at the same time as the development of the back end (slightly following its development), code was not documented and therefore there was a lot of interaction between the developers of the back end and myself.

Four developers and one project manager (who also developed part of the application) where involved. Dr. Petra Zimmermann was responsible for the development of the rich client's graphical user interface. Dr. R. Sai Anand was responsible for the development of the back end (database, business logic and glue logic for the rich client GUI). Nick Nabholz developed a concept for distributed work with ShaRef bibliographies and an implementation as a plug-in for the application [2]. My task was de development of a Web application and glue to the back end and to Nick Nabholz's plug-in. Dr. Erik Wilde managed the project and supervised me during this master's thesis.

A paper describing the software's use cases [4] existed and helped me understand the big picture and the operational area of the Web interface.

A brief description of the architecture makes it easier to understand the following steps: ShaRef is a client/server application based on the Java programming language using Java's Remote Method Invocation (RMI) for the communication between client and server. The (rich) client is installed locally on a user's machine and can be operated stand-alone (it comes with a built-in database).

But one of the main goals of the ShaRef project is to allow users to share (and publish) their bibliographies. For this purpose the ShaRef rich client can be synchronized with the ShaRef server which resides at ETH and stores all the bibliography information in a centralized manner.

Another possibility to access the data stored on the server is the Web interface which was created during this work. It runs on a separate Web server and also accesses the ShaRef server by using RMI.

Evaluate technologies and check alternatives

Knowing this, in a second step I had to evaluate a technology on which the Web application would be based. The evaluation process involved sighting and evaluating actual, older and future Java Web Technologies and comparing them, always keeping the goal of this master's thesis in mind. It was important to find a technology which was strong in building practical graphical user interfaces in Web environments. Furthermore, the rough interface of the business logic existed and the business logic was separated from the Web application, which helped me in making some decisions. For more information on the choice of the technology see chapter 2.1.

Build the system

The third step was mainly programming the Web application. As soon as a feature was added in the back end, I read the code and made a concept to glue the Web application to it.

During this step extensive brainwork was made by all the team members and minor changes to the database schema (and other parts of the system) had to be reflected the back end code, and the two GUIs.

Document and prepare for delivery

In a forth step the documentation had to be written. We used Erik Wilde's HSG system[1] [3] to create a HTML help system for a user documentation and a technical documentation. Since the ShaRef project is only running until the end of the year (and then will be passed to another, at this point unknown department) the technical documentation must be very accurate to allow further maintenance and extensions of the system.

[1] http://dret.net/projects/hsg/

Chapter 2

Knowledge and Work

2.1 Evaluation of technologies

2.1.1 Java framework

After evaluating different Java frameworks for building Web applications and writing an article on this topic in Heise's iX magazine [1] I decided to use a standard which was just evolving in the Java Community called *JavaServer Faces* or JSF.

JSF is a framework based on the Servlet technology which strictly uses the model-view-controller pattern. It was created in the Java Community Process (mainly supported by SUN microsystems) and its first reference implementation (RI) was also written by that company. The Apache Community then wrote an open source implementation called *MyFaces* which is currently in version 1.1.0 but still under development to match the specification. MyFaces also added some prebuilt components called *Tomahawk* (JSF was built with a focus on extensibility) which could be used by JSF implementors (and some of them also found their usage in ShaRefWeb). Other companies are also integrating JSF and adding components in their products (like Oracle in JDeveloper 10).

2.1.2 Infrastructure

To use a servlet-based technology like JSF you need a Web server which is capable of acting as a servlet container. In some lectures I have had during my studies we had used Apache's Tomcat Servlet Container, so I decided to stick to what I know and installed the latest version 5.5 of Tomcat.

The Web application is deployed as a WAR file and should be running on every J2EE-capable servlet container (like Jetty, JBOSS, WebLogic, WebSphere etc.). Some code though is Windows specific (accessing configuration files and XSLT style sheets in the file system) so the server's operating system should be Windows (any release). All the configuration files, necessary JARs and tag libraries are included in the WAR so the entire application lies within one file.

The application of the code was mainly done on the Eclipse IDE (Versions 3.0 and 3.1), some little code snippets were created on Oracle's JDeveloper 10g J2EE edition.

2.2 Experiences, Teamwork, Lessons learnt

During my mandatory industry internship for my ETH studies I have worked in a research company where I had mainly worked alone. One reason for me to choose this master's thesis was the fact that I could work in a team and gain experience as a team programmer and member.

2.2.1 Concurrent Programming

So probably the biggest lessons I learnt were those somehow connected to engineering in a team. Creating a second GUI to a client/server application during its development is almost like adding a feature to a legacy system (with some additions that make life harder). Most (almost all) of the code is not commented and class members can change after each meeting, making interfaces change.

Comment code when it's created

A good advice to teams programming concurrently is to comment code right after it is written. This makes it much easier for teammates to use the interfaces and the newly created code. ShaRef is heavily XML-oriented which means that the object structure depicts the XML schema of the database. This makes it more complicated to program because sometimes the usage of the schema is considered more important/practical than the usual practices of object oriented design. This makes knowing the base schema very important. A lot of questions and problems can be omitted when the database and object structure is well understood.

Separate the view from the logic

When creating a business logic module try to keep the view (GUI elements) seperated from the logic. Using interfaces with the different GUI implementations makes life easier for the creator of the second GUI (which in this case was me). JSF in itself is supporting that fact by forcing (and supporting) the user to implement the MVC pattern.

Learn CVS and use it wisely

CVS is a very simple yet complicated protocol. Branching and versioning etc. makes it hard to re-merge and integrate code that has been separated from the other code. Whenever possible only use one CVS repository and one location for the team even when classes are heavily edited by multiple programmers (sounds easy).

Use a bug tracking system

It's a good idea to use a bug tracking system to standardize and structure issue resolving in the team. We used the web-based bug tracking system Mantis[1] which is easy to understand and handle.

2.2.2 Programming and Java Technologies

Of course I improved my skills with the Java programming language and its APIs. I especially learned to encapsulate Modules with JavaBeans, learned advanced concepts at manipulating Strings and files and in using RMI and XML APIs. For example, programming the editor parser (the module that reads the strings entered by users and transforming them into XML-structured objects) was sometimes very challenging and needed a lot of brainwork (see the technical documentation).

Sevlets and JSP

I learned the concept and programming of Java servlets and JSP (JavaServer Pages) based Web applications. JSF pages are basically JSP pages (they have the *JSP* filename suffix) using special tag libraries. The entire JSF system is based on servlet technology and therefore relies on a servlet basis.

JSF - a new technology

I learnt to use a new Web Framework (JSF) and saw how a large-scale open-source project (MyFaces) evolves (I subscribed to the MyFaces newsgroup during the entire development process).

Tomcat

For the first time I uploaded files and accessed the file system out of the Tomcat context. Uploading in ShaRefWeb is done by using a MyFaces component called *fileUpload*. The application has to reach out of the JSF framework and access the Servlet context to access the file system of the computer.

Eclipse IDE 3.0 and 3.1

I developed the application on Mac OS X (until Java 5 was used; OS X does not support Java 5 at this point) and later Windows XP Professional. On both machines I used the Eclipse versions 3.0 and later 3.1.

For smoother engineering with Eclipse I installed the *Genady RMI plugin*, the *MyEclipse enterprise workbench plugin* and the *oXygen XML editor (Eclipse plugin version)*.

[1] http://www.mantisbt.org

The RMI plugin supports an Eclipse developer in creating RMI based applications. It generates, for example, RMI stubs automatically (after activating that functionality) for all classes using RMI.

The MyEclipse plugin supports a user in creating (among others) J2EE Web projects. The version I used also has JSF support built in and a (not yet very sophisticated) WYSIWYG editor for JSP pages. It supports the user in deploying Web projects to the most common servlet containers (like Tomcat).

The oXygen plugin is an Eclipse version of the full oXygen XML editor. It supports the user in writing/editing/reading XML files, Schemas, XSLT stylesheets etc.

Debugging a Web application

Debugging a Web application is not easy and initially puts a programmer back in time to when debugging meant using *print* functions... If the support for JSF-debugging is not included in the used IDE, the process of debugging involves:

1. Writing code and pages (including creating *FacesMessages* with the stacktrace of exceptions)
2. Stopping the Servlet container / Web server
3. Packing and deploying the updated Web application
4. Restarting the Servlet container / Web server
5. Calling the desired page in a browser
6. Calling the desired function (e.g. by entering text and klicking on a button) and read the possible error messages.

This consumes a lot of time, is somewhat cumbersome and it is not possible to use a debugger for direct variable access, code linking etc.

The following improvements should make life a little easier for web-developers:

- Cut the login functionality during debugging. It takes a long time to login every time just to see if a picture is displayed correctly. When programming a login functionality include a switch to turn it off.
- Use an IDE which supports WYSISYG (*what you see is what you get* preview) for the JSF pages. For basic Web design and displaying new components it shouldn't be necessary to deploy the application after each change.

After tampering around with these inefficient workflows, I used a feature provided by the *myEclipse* Eclipse plugin which actually links code to the debug mode of the Servlet container. This means that (after solving some problems with help from Web forums and mailing lists) I could almost debug the Web application like normal code. If you ever program a web project, get a supporting module that enables this, this feature is invaluable!

2.2.3 Other intellectual contributions

Apart of the ideas I described in other chapters of this report and the code itself my contributions are mainly apparent in the glue logic.

Usually JSF pages access JavaBeans[2] to display dynamic data. At the end, all Information from the business logic is dispayed as strings on the JSF page. So the glue logic had to fetch the objects in their schema structure and extract all relevant information in nice looking and intuitively readable Strings. Multiple entries (e.g. author information in the reference editor) hat to be merged. This means that before displaying the information that has been gathered, there is some amount of string manipulation that has to be done. For details please have a look at the JavaDoc documentation for the code.

The other way around it's even more complicated: Strings have to be read (they are hand typed by users, so they can have arbitrary characters) from the JSF page, parsed in a smart way and then objects have to be created according to the contents. I would like to describe one particular problem to clarify the issue:

A document that is referenced in ShaRef can have multiple primary authors. Since collaborating scientists (who are the main focus group of ShaRef) often come from different countries, they can have very different name structures. People from the Netherlands for example may have a van *(we call it link) in front of the family name, people from Spain usually have two family names, family dinasties may include a* Jr *(which we call lineage). So it is nearly impossible to distinguish* Vincent Van Gogh *from* Freddie Prince Jr. *or* Apple (Switzerland) Inc. *when parsing a string. Users must follow some rules when entering Author names in the ShaRef system, e.g. connect two family names with a dash (*Maria Martinez-Sanchez*). The order has to be standardized, too. In ShaRef it's First name, Family name, Link, Lineage (without the commas).*

2.2.4 Project Management and Circumstances

During the project there where some factors making live easier and some making life harder for me as the thesis writer.

First I state the points that could be improved from my point of view: The structure of the project was sometimes hard to understand. Many discussions changed schedules, first adding features to the product, and towards the end removing features due to time issues. A clear milestone plan and decisions upon approaching those would probably help in understanding each others goals better.

Meetings were sometimes cancelled without noticing the participants. Especially meetings at the start or end of the day may involve changing plans (which may involve other people). A note, saying that the official day starts an hour later would surely put a smile on people's faces ;).

The project was handled very scientifictly. Sometimes very complex thoughts would generate difficult coding which may have been replaced by more practical solutions in the industry.

[2] http://java.sun.com/products/javabeans/docs/spec.html

The amount of project and product information reaching me at the beginning was huge. I had to start filter mails that were important for the project but reached me at a to early stage. The other team members had a common vocabulary which I did not understand from the start (by the way, ShaRef helps support situations like this one) and I had to find out what was important for me and what wasn't (at this stage).

Here some points which made the project a success from my personal point of view: The climate during the project was very good for me, it was a pleasure to work for the project manager and with the team-mates. We had some very interesting discussions and also spent free time together which makes working together easier.

The project manager and all the members were very open for input and discussions. In an environment like this, the knowledge and creativity of many people can merge and create a powerful product.

I could manage a lot of my contribution to the project myself. Most of the time I worked by myself and only interacted (in business-relevant issues) with the other mates when necessary. Working in my own pace and setting my goals increased productivity. Of course this kind of work involves trust by the project manager and I got his trust throughout the entire thesis.

Writing the article for the iX magazine [1] was really exciting. I am very proud of that article and will consider to write more articles during my future work.

Chapter 3

Results

3.1 The Web application - Documentation

The application is documented within three separate documentations, the ShaRef Help System.

The first two parts are created with Dr. Erik Wilde's HSG system and are nested hierarchically in the main help system. The following Lists show the contents:

1. ShaRefWeb Help - User Documentation
    - Login Page
    - Register Page
    - Main Page
        - Workspaces/Reference lists Pane
        - Import Pane
        - Export Pane
        - Messages Pane
        - Editor Pane
        - Commit Page
2. ShaRefWeb Help - Technical Documentation
    - Session Management
    - Workspaces, Bibliographies and References
    - Editor Workflow
    - Import and File Manipulation
    - Export and Publishing Management
    - Commiting and Database Manipulation

3. ShaRefWeb - JavaDoc

    - The standard JavaDoc system was used to comment all Java Files used in the project. Probably this will be integrated to the complete ShaRef JavaDoc repository when the project finishes.

Please read these documents for further information on the technical and usability issues.

3.2 Features to be added after completion of the master's thesis

As I mentioned above, during the development of Web client and the main application a lot of brainwork was done by all team members. Interfaces changed a lot and new ideas where raised at almost every team meeting. During the master's thesis the foundation for ShaRefWeb was laid. After completing this master's thesis I will remain in the project team and add features to the Web client:

3.2.1 Sort, Search, Mark

The workspace listing the references will be sortable by clicking on the table headers, similar to the rich client. This feature is included in a Tomahawk-component which extends JSF's *dataTable*.

Search on references has to be implemented manually by using the Workspace and Reference objects. It is already implemented on the rich client and the Web client can use a lot of functionality.

At this point only entire workspaces can be selected (e.g. for commiting). The Workspace classes contain fields and methods to mark Workspace members and they will be supported in release 2 of the Web client.

3.2.2 Keywords (Definitions) and Associations (Definitions)

A support for managing, editing and adding keywords and associations will be implemented. It will consist of displays to list the elements and editors to add/change them. This will also affect the usage and design of the rich text editor (see below).

3.2.3 Group management

Some features of Nick Nabholz's plugin are already implemented (like Session Management, Login, Access rights for bibliographies) but the group management part is not yet supported. It has to be implemented completely in the back end first.

3.2.4 Messaging

Messaging is an important concept in distributed work environments. When the group plugin is fully integrated, ShaRef users will be able to send messages (possibly containing attachements like references) to each other, selected groups, bibliography administrators, external users (by e-mail) etc.

It should be possible to attach entire workspaces and maybe exported documents to the messages. For external mailing some kind of SMTP server has to be installed and configured.

3.2.5 Rich Text Editor

The rich text editor allows editing of annotations and abstracts attached to references. Decisions on the layout still have to be made. The rich text is formatted and stored as XML with embedded links, cross-references to other references and keywords attached to it. The possibilities are:

- Display a text editor and let the user edit the XML directly (error-prone).
- Display a WYSIWYG editor (e.g. Word) and update the according XML in the background
- Display a split screen with both editors shown and one or both editable
- Display a WIKI-like editor (enter markup by entering special characters or character combinations).

In all cases the user should be supported in entering tags and/or formatting text and syntax checking should be made to validate the resulting XML.

3.2.6 Multiple languages

Java implements the concept of message bundles and locales to create programs that support multiple languages. The first languages that will be supported are English as default and German as additional language. The bundle concept makes it very easy to add other languages.

3.3 Thanks

I would like to thank everybody who supported me directly or indirectly during this master's thesis and my studies at ETH, especially the people involved in the ShaRef project (Erik, Petra, Sai, Nick and Till), and the people supporting me behind the scenes (especially my mother and Claudia), the boys, and a lot of other important people (You know who you are!).

Bibliography

[1] Thierry Bücheler, Erik Wilde: *Rahmenentscheidung - JSF versus Struts: Serverseitige Java-Techniken im Web*, iX-Magazin Oktober 2005, p.133

[2] Nick Nabholz: *Ein Benutzerkonzept für kollaborative Applikationen am Beispiel von ShaRef*, Diploma thesis at the Hochschule fr Wirtschaft, Verwaltung und Technik Zürich, September 2005

[3] Erik Wilde: *Augmenting XHTML for Help and Documentation*, International Conference on Intelligent Agents, Web Technology and Internet Commerce (IAWTIC 2005), Vienna, Austria, November 2005

[4] Erik Wilde, Sai Anand, Petra Zimmermann: *ShaRef Use Cases*, Internal Paper of the ShaRef project team, April 2005

[5] Erik Wilde, Sai Anand, and Petra Zimmermann: *Management and Sharing of Bibliographies*, 9th European Conference on Research and Advanced Technology for Digital Libraries (ECDL 2005), Vienna, Austria, September 2005.

[6] Karl Rehrl: *Auf Stütze; Jakarta Struts - Hilfe für komplexe Webanwendungen*, iX-Magazin November 2002, p.130

[7] Rainer Klute: *Mit neuem Gesicht; Java Server Faces vereinfachen Webentwicklung*, iX-Magazin November 2004, p.128

[8] Boudigue Alioum, Tilmann Kuhn: *Entwicklung von Webapplikationen mit den Struts und Expresso Frameworks - ein Vergleich*, seminar thesis, University of Karlsruhe, Germany

[9] Erik Wilde: *Towards Conceptual Modeling for XML*, Berliner XML Tage 2005 (BXML 2005), Berlin, Germany, September 2005.

[10] Erik Wilde: *Shared Bibliographies as Hypertext*, Technical Report TIK Report No. 224, Computer Engineering and Networks Laboratory (TIK), ETH Zürich, May 2005.

[11] Sai Anand and Erik Wilde: *Mapping XML Instances*, Fourteenth International World Wide Web Conference (WWW2005), Shiba, Japan, May 2005.

[12] Erik Wilde, Sai Anand, and Petra Zimmermann: *ShaRef: XML-Centric Software Design*, Technical Report TIK Report No. 213, Computer Engineering and Networks Laboratory (TIK), ETH Zürich, February 2005.

[13] Erik Wilde: *References as Knowledge Management*, Issues in Science Technology Librarianship, No. 41, Fall 2004.

[14] Erik Wilde: *A Tool for Bibliography Management and Sharing: The ShaRef Project*, D-Lib Magazine, 10(9), September 2004.

[15] Erik Wilde: *Usage and Management of Collections of References*, Technical Report TIK Report No. 194, Computer Engineering and Networks Laboratory (TIK), ETH Zürich, June 2004.

[16] Erik Wilde: *Towards Federated Referatories*, SINN03 Conference on Worldwide Coherent Workforce and Satisfied Users, Oldenburg, Germany, September 2003.

[17] Qusay H. Mahmoud: *Servlets and JSP Pages Best Practices*, article on http://developers.sun.com, 2003

[18] Marc Hedlund: *Why JSP Sucks So Hard*, discussion article on http://oreillynet.com, 2002

[19] Janice J. Heiss: *JavaServer Faces and Java Studio Creator: The Evolution of Web Application Frameworks*, article on http://developers.sun.com, December 2004

[20] Govind Seshadri: *Understanding JavaServer Pages Model 2 architecture*, article on http://www.javaworld.com

[21] R. Johnson: *Existierende Systeme I: Bibliotheken und Frameworks*, seminar thesis, University of Paderborn, Germany

[22] Christian Schneider: *Frameworks = (Components + Patterns)*, Communication of the ACM, October 1997/Vol. 40, No. 10

[23] A. Bien: *J2EE Patterns*, Addison-Wesley, ISBN 382731903X