Seminar Paper

Dictionary Software based on Ajax Frameworks

Matthias Kerstner

Institute for Information Systems and Computer Media

Graz University of Technology

Head: Herman Maurer O.Univ.-Prof. Dr.Dr.h.c.mult.

Supervisor: Denis Helic Dipl.-Ing. Dr.techn.

September 2007

Abstract

The necessity for dictionaries cannot be disputed. Although there are a vast variety of them they all follow a basic schema – a key-value combination. It is interesting to see that many of the well known dictionary producers recently also provide electronic versions of their products, ranging from CDROMs over USB media to online databases. With the event of Ajax, web-based applications can be turned into desktop-like environments. It provides developers with an effective way to build web-applications using existing facilities (browsers, etc.), while still profiting from the latest technology.

Contents

List of Figures

1. Introduction

As many traditional fields of science have already moved towards using computer systems to provide access to relevant data, many of the popular dictionary producers like Oxford[1], Webster's[2], Langenscheidt[3] and Duden[4] today also provide software-based versions of their products. Most probably one of the best known online dictionaries is Wiktionary[5].

In the course of planning to develop the dictionary software "Wörterwelt" the following main criteria were kept in mind:

- No need for any extra software (installation)
- High accessibility
- Interactivity
- Expandability (new language translations, etc.)

High attention was paid to the ability to run it inside browsers. Using this approach not only would it be possible to use a centralized data storage (i.e. a database), but also since the majority of the current operating systems support some form of (graphical) browser software no special installation procedure would be needed.

When it comes to interactivity inside browsers the main keyword nowadays is "*Ajax*". Ajax incorporates many different technologies to be able to realize desktop-like web applications [1]. In recent times the basic technology behind

[1] Oxford University Press, http://www.askoxford.com/?view=uk

[2] Webster's Revised Unabridged Dictionary, http://machaut.uchicago.edu/websters

[3] Langenscheidt, http://www.langenscheidt.de/b2b/online-woerterbuecher_143.html

[4] Duden, http://www.duden-suche.de/suche/

[5] Wiktionary, http://en.wiktionary.org/wiki/Wiktionary:Main_Page

Ajax has experienced a revival [2][6] and a wide range of Ajax frameworks have been developed.

In the following we will first be shortly discussing dictionaries in general, while creating a bridge towards software solutions using Ajax frameworks. The second chapter will then be dealing with some of the most popular Ajax frameworks currently available. We will be comparing their architectures while keeping the focus on Java based Ajax platforms. Finally you will be presented the framework chosen to build the dictionary software “Wörterwelt”.

[6] Refer to chapter 3.1 for further information on Ajax

2. Dictionaries

[3] Defines a dictionary as follows:

> *"A reference book containing words usually alphabetically arranged along with information about their forms, pronunciations, functions, etymologies, meanings, and syntactical and idiomatic uses."*

Although there are various types of dictionaries they basically all consist of tuples of information: the search term "*K*" (key) and the information attached to this search term - the actual content "*V*" (value). As to say K *maps* onto V, as shown in Fig 2-1.

$$K \longrightarrow V$$

Figure 2-1: Key-value mapping

Due to their nature dictionaries can be easily compared to software databases. As computer systems have become more powerful by using fast search algorithms and efficient data structures [4], searching for a key in a given set is generally not a big issue anymore. Furthermore all major RDBMS systems like Oracle, MySql, etc. provide very efficient search functionality to achieve this mapping.

The task of searching in dictionary software is (heavily) interactive (e.g. entering the search term, submitting the data, browsing through the result set, marking a result entry, etc.) adequate facilities should be provided to the user. Due to the fact that we were looking for a browser-based solution "Wörterwelt" Ajax was the way to go.

The next chapter will give you a quick overview of Ajax itself, followed by a comparison of some of the currently most popular frameworks.

3. Ajax Frameworks

> "The Web wasn't ever as functional or useful as client software, and Ajax just knocks that ball out of the park." [5]

What started as a simple *XmlHttpRequest* object introduced by Microsoft in its IE [6], nowadays represents a big field of interest – Ajax technology. Ajax has become very popular due to its ability to free browsers from their static life by providing a great amount of new interactivity features (a.k.a. "*widgets*"), therefore enriching web applications.

As a reaction to the "Ajax-hype" a great amount of frameworks have been developed. They basically all follow the same idea: Providing APIs so that developers don't need to fiddle around with browser-dependent JavaScript issues.

First of all we have to differentiate between the two essentially different types of frameworks. On the one hand there are the ones that can be run by simply integrating JavaScript files in web applications (e.g. *script.aculo.us*[7]), and on the other those that need additional servlet containers (e.g. *ZK*, *DWR*). While the first option often is the faster way to go, the second one offers much more flexibility.

JavaScript-only based frameworks like *script.aculo.us* are only executed on the client side, updating DOM entities locally. Working with frameworks that are based on additional servlet containers local events, triggered by users, are generally directed to the servlet container, which processes the request and sends back a response to the user, causing the actual local DOM update. Therefore these types of frameworks can also be compared to conventional RPC mechanisms such as SOAP[8] or RMI[9], with the benefit that they run over the web without requiring web-browser plug-ins [7].

---

[7] http://script.aculo.us/

[8] http://www.w3.org/TR/soap/

[9] http://java.sun.com/javase/technologies/core/basic/rmi/

Due to the vast amount of frameworks available nowadays [8] yearly conducts surveys of the most popular ones. Based on the survey data for 2007 [9] produced a chart showing the most popular Java/Ajax frameworks, which is shown in Fig 3-1.

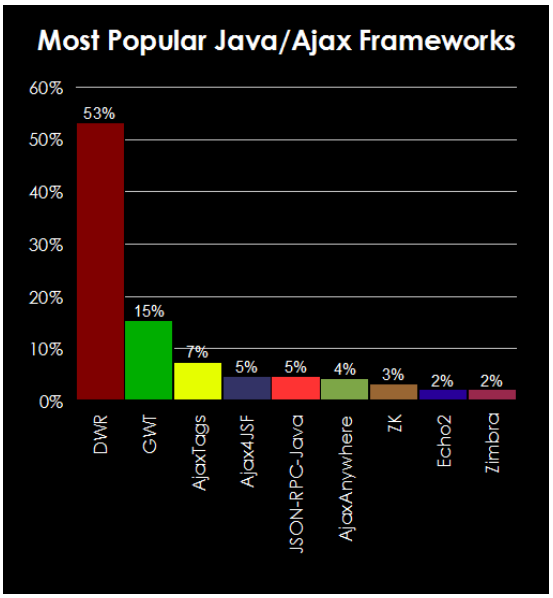

Figure 3-1: Most popular Java/Ajax Frameworks

The last part of this chapter will deal with a short overview of the Ajax technology, followed by a comparison of some of the most popular Java/Ajax frameworks according to [8]. Although there are various different Ajax platforms, as shown in Fig 3-2, this paper only discusses the Java-based ones listed in Fig 3-1. See [10] and [11] for a more complete list of other frameworks for different platforms.

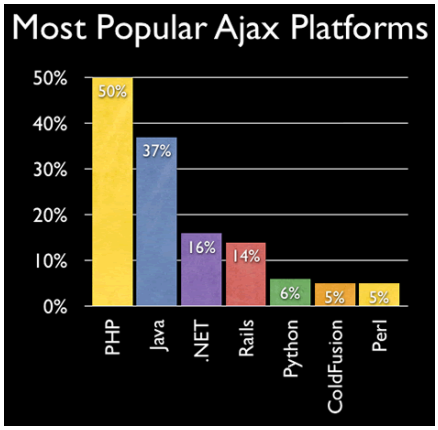

Figure 3-2: Most popular Ajax Platforms

3.1. Ajax Overview

Ajax (Asynchronous JavaScript and XML) is basically a new buzzword introduced by Jesse James Garrett [12], for a technology that has already existed for quite some time [13]. Ajax itself incorporates two powerful browser techniques that were simply not noticed by web developers, until companies such as Google started to provide services like Google Mail[10]/Maps[11]/ Suggest[12] [14].

These two techniques basically consist of

- Sending requests to servers without completely re-loading the current website (*asynchronous transmission*)
- Parsing/Manipulating XML documents

This concept of asynchronous transmission is what makes Ajax so powerful. It is now possible to send HTTP requests from within a website without having to re-request the entire page, thus freeing the browser from its page-based model, while additionally saving bandwidth [15]. Furthermore the executing browser software is not halted during the server-client interaction, as it does no longer need to wait for the HTTP transmission to return. Using this approach it is possible to create widgets that act as if they had been only executed on the client side.

In the following we will briefly discuss some of the most popular Java/Ajax frameworks, as shown in Fig 3-1.

3.2. DWR

DWR (Direct Web Remoting) is according to [8] currently by far the most popular framework. It consists of two main parts: a Java servlet on the server and JavaScript files on the client side. DWR basically works by using

[10] http://mail.google.com/mail/help/intl/de/about.html

[11] http://maps.google.com/

[12] http://www.google.com/webhp?complete=1&hl=en

dynamically generated JavaScript (“event-handlers”) based on Java code [7]. In order for interaction to occur between the client and the server side a call-back function must be provided when calling the respective JavaScript code. When the server side execution terminates, this call-back function is called, updating the client side, as shown in Fig. 3-3.

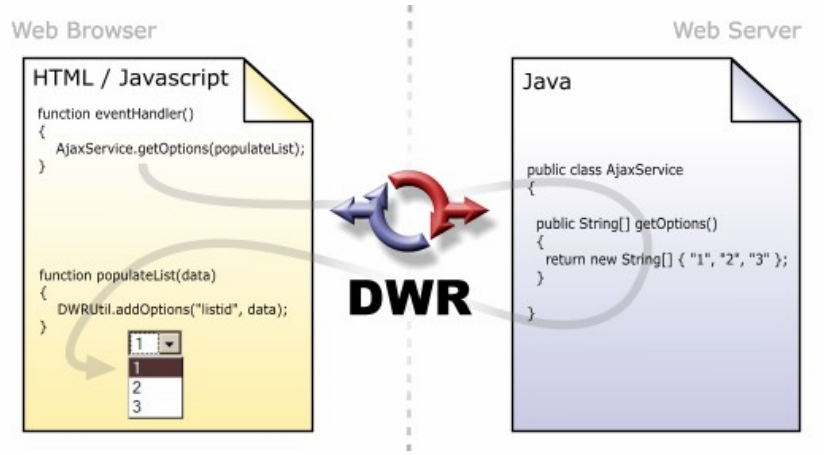

Figure 3-3: DWR asynchronous client/server interaction

3.3. GWT

GWT (Google Web Toolkit) most probably became known due to Google’s Mail/Map/Suggest applications. In contrast to the other frameworks available GWT can be run in two different modes: *Hosted* and *Web* mode [16]. These modes can be seen as consequent stages in the deployment process. Application in Hosted mode are run locally within the current Java Virtual Machine (using GWT’s own “hosted” web browser), whereas in Web mode they are run as JavaScript embedded in HTML code.

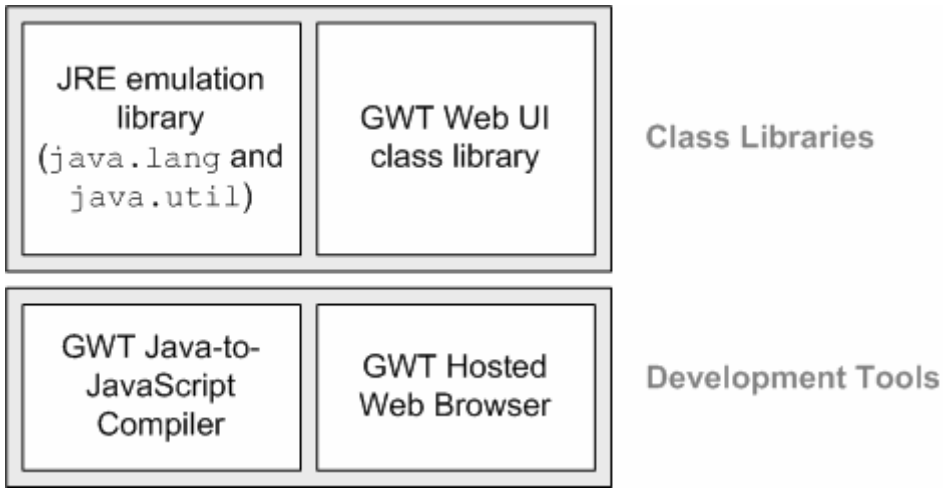

Figure 3-4: GWT architecture

To compile Java files for Web mode GWT uses its own *Java-to-JavaScript compiler*. The resulting JavaScript code can then be included in HTML files that will actually be displayed to the user.

As shown in Fig 3-4 GWT's principal architecture consists of four components that are separated into two sub-categories. The *Class Libraries* are basically pre-compiled Java classes. They represent the most common Java standard classes (e.g. *java.lang*) on the one hand and the most widely used widgets on the other. The *Deployment Tools* make up the second sub-category. These are used to compile Java to JavaScript code (GWT Java-to-JavaScript Compiler) and to run applications in Hosted mode (GWT Hosted Web Browser).

3.4. AjaxTags

According to [17] *AjaxTags* is basically a collection of JSP tags that should simplify the way how web developers achieve (client-side) *web-form* effects, without the need to write any JavaScript code. AjaxTags therefore provides developers with tags that can be integrated in JSP to create Ajax-capable web-forms. A tag itself is made up of a set of parameters defining the form-component's behaviour.

Unfortunately it currently only offers a very limited amount of form-update mechanisms. These range from auto completion based on character input to

an input field over callout or balloon popups for highlighting content to toggling images and form field states.

3.5. Ajax4JSF

Ajax4JSF (Ajax for JavaServer Faces) is an open source framework that provides a so-called "*skinnability feature"* [18]. Using this feature it's much easier to manage the LAF (look-and-feel) of web-based applications.

Ajax4JSF itself is the basis for so-called *JavaServer Faces* frameworks. Different component libraries such as RichFaces[13] are enhancing these frameworks by providing additional widgets. For example RichFaces enriches Ajax4JSF in two ways. First it adds and enhances visual components and secondly it implements Ajax4JSF's skinnability feature, while additionally providing a large number of predefined skins [18].

3.6. Echo2

As shown in Fig 3-5 the *Echo2* framework is divided into three components. The *Application* "layer" provides the primary API for developers to change the application's state [19]. What makes this layer special is the fact that it does *not* provide any means of communication between the client side and Echo2's actual rendering mechanism. Therefore the primary API does not provide direct rendering functionality for UI components. According to [19] this decision is based on the idea that it's now possible to use the Application module in scenarios where the UI was rendered to for example SWING[14]. The actual UI updates are communicated through Echo2's *Update Manager*.

Echo2's *Web-Rendering-Engine* on the other hand consists of a client and a server part (Client/Server Engine). It provides a set of tools that accomplish the client/server interaction. The server-side portion is the connection to Echo2's third layer – the *JEE*.

---

[13] http://labs.jboss.com/jbossrichfaces/

[14] http://java.sun.com/javase/technologies/desktop/

The Server Engine provides an HTTPServlet that processes requests from the client side by delegating it to an appropriate Service object [19]. The Client Engine on the other hand represents a JavaScript application running on the client web-browser. In case of user actions communication between the Client and the Server Engine is established through the well-known *XMLHttpRequest*.

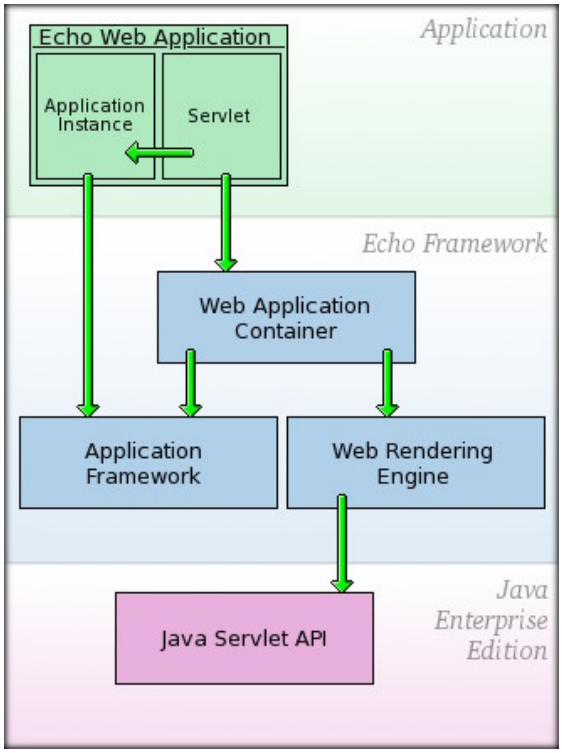

Figure 3-5: Echo2 architecture

4. ZK

ZK basically consists of three components which then again are separated into a client and a server part. As shown in Fig 4-1 the client side consists of the *ZK Client Engine,* whereas the server portion is made up of the *ZK Loader* and the *ZK AU Engine*.

ZK pages are written in a special language called *ZUML,* which can be compared to HTML. ZUML provides tags to describe what components to create and furthermore how they should be visually represented. Once created these components remain valid until the current session expires (i.e. the user closes the browser, etc.).

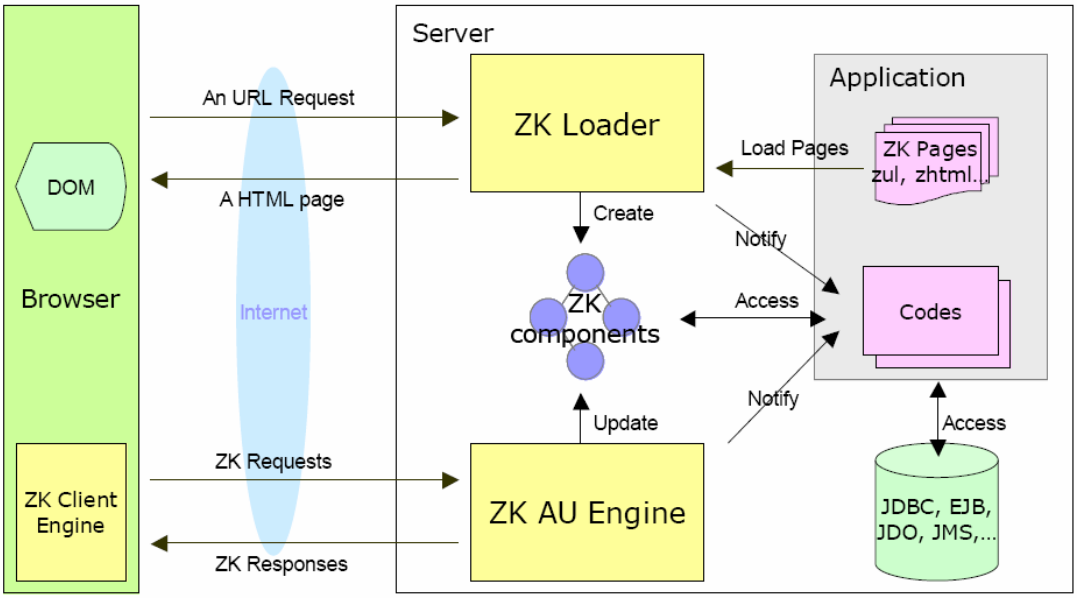

Figure 4-1: ZK architecture

The flow of events starts with a user requesting a ZK page. Upon this request the ZK Loader loads the requested page, interprets it and renders the result into HTML pages [20]. The interaction between the client and the server side is achieved by information exchange between the ZK Client Engine and the ZK AU Engine. They follow the event-driven programming model by delivering events triggered on the client side to the application running on the

server. The server portion in return responses by requesting an update of the client's DOM tree through the Client Engine, corresponding to the changes done to the application [20].

4.1. Why ZK

ZK was chosen to be used for implementing the dictionary software "Wörterwelt" due to the following five reasons:

1. It follows a very strict component structuring schema
2. ZK components are completely based upon Java classes, adding or changing components can be easily achieved by implementing the appropriate interfaces
3. Support for scripting-codes and EL-expressions
4. Great support and documentation
5. Independent development branch for mobile devices, etc.

ZK's underlying component structure starts by differentiating between a *Desktop* and *Page*. In ZK it's possible to include a ZUML page inside another, while both are serving the same Uri. This means that if the parent Page is requested the included one too is created by the ZK Loader. Due to this the concept the Desktop component was introduced. It serves as a container for Pages belonging to certain applications. This makes it possible to interactively add/delete Pages (or any other components) from a Desktop [20].

Since ZK components are entirely written in Java adding and changing components is rather simple. Components follow interface declarations that must be implemented by developers. Thus substituting components of the same type can be easily done.

Although writing Java classes is the more rigid way to go when building (bigger) Ajax applications ZK also supports the use of scripting-code and EL-expressions inside ZUML pages. Currently supported scripting-code languages range from Java to Groovy and more are still to come. But unlike

JavaScript for example ZK executes these codes on the *server* side, therefore relieving the client's browser [20].

One of the most common problems using emerging frameworks is the lack of documentation. Unlike many other well known frameworks ZK provides comprehensive and up-to-date documentation and great user support through various forums at the project site. Furthermore feature requests are processed very quickly.

Another point to be made is ZK's separate development branch for mobile devices – *ZK Mobile*. This compact framework was specifically developed to be used on mobile devices and currently offers a handful of *MIL* (Mobile Interactive Language) components [21].

Summing it up ZK provides a very rigid way to write Ajax-based web applications. It is based on a clean architecture, with simplicity kept in mind. Apart from the actual framework the ZK project additionally offers smaller sub-projects, such as ZK Mobile, making it a good choice even for special purposes.

Conclusion

Ajax certainly opens web-developers new perspectives. Using Ajax's underlying mechanism for asynchronous transmission it is possible to develop highly interactive, desktop-like web-applications. Various frameworks already exist, encapsulating browser-dependent JavaScript code. While client-side-only frameworks are often the faster way to go, servlet-based ones offer more flexibility.

While comparing the currently most popular Java/Ajax frameworks, *ZK* has proven to be a good choice for the dictionary software "Wörterwelt". Although many of the frameworks provide the same basic set of widgets ZK was taken in favour of them, due to its features, ranging from its clear component structuring schema, over scripting-code and EL support, to even separate development-branches for mobile devices.

Bibliography

[1] http://www.ajax-community.de/allgemein/2921-kleiner-einblick-ajax.html (28.02.2006)

[2] Woolston, D.: “Pro Ajax and the .NET 2.0 Platform”; Apress, USA (2006)

[3] Merriam-Webster: “Webster’s Third New International Dictionary”; Unabridged, USA (2002)

[4] Black, P. E.: “Dictionary of Algorithms and Data Structures”; September 1998, http://www.nist.gov/dads/, (05.09.2007, NIST)

[5] LaMonica, M.: “Ajax gives software a fresh look”; October 2005, http://news.com.com/2100-1007-5886709.html?tag=tb (06.09.2007, CNET News.com)

[6] http://www.ajaxpatterns.org/wiki/index.php?title=XMLHttpRequest_Call (02.09.2007)

[7] http://getahead.org/dwr/overview/dwr (06.09.2007, GetAhead)

[8] http://ajaxian.com/archives/ajaxiancom-2006-survey-results (23.09.2005, Ajaxian.com)

[9] http://getahead.org/blog/joe/2006/09/27/most_popular_java_ajax_frameworks.html (27.09.2006, GetAhead)

[10] http://www.ajax-info.de/uebersicht-ajax-frameworks (04.09.2007, Ajax-Info.de)

[11] http://ajaxpatterns.org/wiki/index.php?title=AJAXFrameworks (12.09.2007, Ajax Patterns)

[12] Garret, J. J.: “Ajax: A New Approach to Web Applications”; February 2005, http://www.adaptivepath.com/ideas/essays/archives/000385.php (10.09.2007, Adaptive Path)

[13] Koujala S.: “AJAX – Concepts Originates from Microsoft”; October 2005, http://news.com.com/AJAX+gives+software+a+fresh+look/5208-1007_3-0.html?forumID=1&threadID=10087&messageID=75175&start=0 (08.09.2007, CNET News.com)

[14] http://developer.mozilla.org/de/docs/AJAX:Getting_Started (08.09.2007)

[15] Heunis, A.: “Why Ajax? The benefits of AJAX explained”; April 2005, http://dotnet.org.za/adam/archive/2005/04/12/17006.aspx (10.09.2007, dotnet.org.za)

[16] http://code.google.com/webtoolkit/overview.html (11.09.2007, Google)

[17] http://ajaxtags.sourceforge.net/ (12.09.2007)

[18] http://labs.jboss.com/file-access/default/members/jbossrichfaces/freezone/docs/devguide/en/html_single/index.html (12.09.2007, Red Hat)

[19] http://www.nextapp.com/platform/echo2/echo/doc/hltov/fundamentals.html (13.09.2007, NextApp)

[20] Potix Corporation: “ZK Developer’s Guide”; September 2007, http://www.zkoss.org/doc/ZK-devguide.pdf (15.09.2007, Potix Corp.)

[21] http://zkoss.org/release/zkmob-rn-0.8.5.dsp (16.09.2007, Potix Corp.)