Separation of Concerns in AJAX-based web applications

– a brief overview about the current situation –

Sven Abels
Research & Development - TIE Nederland B.V.
NL-1119 PS Amsterdam, Netherlands
sven.abels@tieglobal.com

Petra Beenken
Dep. of Computing Science, University of Oldenburg
D-26111 Oldenburg, Germany
petra.beenken@informatik.uni-oldenburg.de

Abstract: AJAX became one of the major topics in the last year. It is often mentioned together with the "web 2.0" concept but is actually independent of it. In this short article, a summary of experiences with AJAX is given. This article concentrates on the situation in web applications that are using AJAX and how they support the separation of business logic and design (i.e. view).

1 Introduction

AJAX is probably one of the most important hypes that have appeared in the last years. Today, the word "AJAX" can be seen everywhere. A study by Gartner recently rated AJAX as one of the five most important technologies in the next two years with high impact for the IT landscape (see [Co06]).

AJAX stands for ***A**synchronous **J**avaScript **a**nd **X**ML*. It describes the idea to change specific parts of a web page without reloading the whole page. AJAX "helps speed up computer operations by cutting down on the need to request fresh Web pages from a distant server computer. Instead, Ajax applications can request smaller chunks of data to update a Web page already on a user's screen." [MB05]. Within the AJAX approach, data is exchanged between the web browser and a remote AJAX-Engine on server side (see [Ga05]). This enables a web application to load and render only a specific part of a web page instead of reloading the whole page in case of updates. Furthermore, updates of those parts are usually performed asynchronously which enables the web application to continue working while the data is loaded from the server side.

The result is a more fluent way to navigate in websites. This makes it possible for websites to behave much more like a desktop application then like a traditional web page. Popular examples of AJAX pages are the Zimbra web mailer and Google Maps.

2 Pros and Cons

It is obvious that AJAX became some sort of a hype, which makes it necessary to carefully judge if the application of AJAX technology is meaningful for a solving a specific problem or not. On the one hand, AJAX enables a smooth navigation and a more fluent way to use a website. This is especially useful for websites that have to deal with a lot of small changes on a web page. It's much more comfortable to simply update the piece of information that has changed rather then to reload the whole site.

On the other hand here are certain disadvantages that come with AJAX. Alex Bosworth has written an article where he describes 17 different problems that he calls "AJAX Mistakes" [Bo06]. The two most important problems that come with the technology are the problematic integration of the back-button functionality and the problem to make the website available for search engines. For example, with the Google AJAX-Framework[1], a whole AJAX application (page) can consist of a single line of HTML-code. All other information and text is integrated into the web page by Javascript when the user looks at the site. This makes it almost impossible to website spiders to catch the content of the page. Hence, the site might not be found in web engines correctly. Because of this it is very important to carefully judge if AJAX is an option for a specific website or not.

Another problem that is especially to be mentioned for business information systems is that AJAX based web sites obviously won't work when switching off Javascript since AJAX is built around this technology. Although companies tend to enable Javascript today, there are still a lot of companies that switched off Javascript support in their web browsers for security reasons.

As mentioned in the last paragraph, AJAX shouldn't be used everywhere. However, there are two good examples for the application of AJAX in the software selling process that should be mentioned here. The first example is the order process. When a customer changes, e.g., the quantity of a product or when he changes the payment method, then it is usually unnecessary to reload the whole web page. Instead of this AJAX could be used to update the price information or add additional information to the order page. For example, Varien has recently shown a video that demonstrated an AJAX-based one-page checkout and ordering process [Va06].

The second example is located in the customer care domain. Many software vendors offer a special login area for the customer where he can download his licenses, updates and receipts. AJAX can be used in this area to make the page as comfortable as possible. One doesn't have to care about the search engine problem, since those pages will usually not be indexed anyway.

---

[1] http://code.google.com/webtoolkit

3 The problem with current AJAX implementations

The problems, mentioned in the last section are more or less technical and might be solved by various technical improvements. Despite those problems, there are some serious concerns with the current direction of implementations found in many AJAX based web applications. From a software engineering point of view one of the most important principles when developing complex web applications is to separate the business logic from the view of a web application.

This ensures that a web application can easily be changed later and it also enables designers to implement a specific visual design independently from the actual business logic. In this specific concern AJAX has, however, lead to a step backwards. The main idea is a very clean and mean approach that fits quite well into modern software development approaches. However, real-world scenarios have shown that AJAX misleads developers to simply add business logic directly to HTML/Javascript code that shouldn't really have to deal with this since it belongs to the "view"-part of a web application. From practical experience, about 80% of today's AJAX projects are including some kind of important business functionality in their Javascript code[2]. We do not state that AJAX necessarily leads to unclean code but the AJAX approach makes it easier for web application developers to not separate between business logic and design. Since AJAX is mostly based on Javascript, it is easy for developers to add some classes to the client-side code in Javascript. This Javascript code is, however, not independent from the view-code any more and therefore we often find a mixture of source code used for rendering a website and of source code, used for calculating business logic related issues.

There are basically two ways to avoid this. One is the application of frameworks that enforce the user to strictly separate between logic and design (this is introduced in the next section). Another one is to carefully avoid a "mixture of concerns". This might for example be performed by running <u>all</u> business logic at the server side or it might be performed by separating the Javascript code into different sections that do not mix each other. There is a general trade-off between the clean separation of concerns and the performance of a distributed system. Placing some uncritical part of the business logic at the client side can reduce the communication between client and server significantly and might often lead to a faster web application. It might in some cases therefore be a sufficient decision to place some minor part of the business logic within the client side of an AJAX based web application but it needs to be ensured that the business part is still separated from the view part. Of course this obviously needs good programmers and a high discipline when designing and implementing an AJAX based web application.

---

[2] Please note that this number is only based on our experience. A formal validation is outstanding in order to get a concrete percentage.

4 Using AJAX frameworks in the development process

It has to be discussed in the last section that developing with AJAX is dangerous because it definitely tends to entrap people to produce unclean code. The reason is that it is very easy to mix Javascript code for business logic and design specific HTML code for presentation instead of separating those concerns. It is therefore meaningful to use a Framework that helps to keep the code clean and to provide a maximum of compatibility for all web browsers. The following list shows an abstract of popular frameworks.

Name	Language	Company	URL
ASP.Net AJAX	ASP.NET	Microsoft	http://atlas.asp.net
Dojo	Javascript	Dojo Foundation	http://dojotoolkit.org
Google WebTk.	Java	Google	http://code.google.com/webtoolkit
Spry	Javascript	Adobe	http://labs.adobe.com/technologies/spry
YUI	Javascript	Yahoo!	http://developer.yahoo.com/yui

Table 1: Brief overview about popular AJAX frameworks

A good example about how a framework can help to support the separation of concerns is the Google Web Toolkit (GWT). In this framework, the entire website is developed in Java, allowing the developers to easily stick to their existing development patterns. After finishing the development, the Java code is automatically converted into corresponding Javascript by the GWT compiler. Hence, a developer does not even have to know about Javascript when developing an AJAX based web application. This approach ensures not only cross-browser compatibility but it also helps to support the separation of concerns. Developers can decide where to place the view and the business logic by simply placing their code in different packages. The `client` package is converted to Javascript and sent to the web browser while the `server` package is used for the server side business logic. This allows the developer to place some business logic on the client side while still separating the view from the logic part (see [GW07]).

References

[Co06] Computerzeitung, Jahrgang 37, Nr. 32-33, Seite 1, 14.08.2006, Gartner, 2006

[Bo06] Bosworth, A.: “Ajax Mistakes”, 2006, available online: http://alexbosworth.backpackit.com/pub/67688, last access: 02.01.2007

[Ga05] Garret, J. J.: Ajax: A New Approach to Web Applications, Essay, Adaptive Path, 2005

[GW07] Google: Google Web Toolkit Developer Guide, 2007, available online: http://code.google.com/webtoolkit/documentation , last access: 03.01.2007

[MB05] Mangalindan, M.; Buckman, R.: New Web-based Technology Draws Applications, Investors, The Wall Street Journal, 03.11.2005, 2005

[Va06] Varien: “AJAX-based One-Page Checkout”, 2006, available online: http://www.varien.com/blog/AJAX-based-one-page-checkout-video last access: 15.10.06