Supervisor:
Li Chen
li.chen@epfl.ch

Professor:
Pearl Pu
pearl.pu@epfl.ch

Semester Project - Final Report

Implementation of an interactive travel map

Joël Schintgen

January 12, 2008

Abstract

The goal of this project was to implement a city-map, in which travel-related information, such as hotels, restaurants, shop and sights, are directly displayed, and thus gives a quick overview of all the relevant locations. Icons on the map also show how this location was rated by other users.

The user sees at first glance how an item was rated by other users. With a single click on the icon he can read a description of the hotel and the reviews and experiences of others.

He has even the possibility to refine his search based on the currently selected item. This example-critiquing makes the map more interactive and guides the users in their information searching process.

Contents

1 Introduction **4**

2 Related work **4**

3 Interface **6**
3.1 The list 7
3.2 The map 8
3.3 Infowindows 8
3.4 Other features 10

4 Implementation **10**
4.1 Collection of information 10
4.1.1 Wrappers 11
4.1.2 Simple form as interface to populate the database . . . 11
4.1.3 Database and Registers 11
4.2 The site class 12
4.3 Setup of the map 13
4.4 Interaction with the server 13
4.5 Getting a list of items 14
4.6 Displaying the items 14
4.7 Getting the details and user reviews 15
4.8 Localisation 15
4.9 Special treatment for Internet Explorer and issues 15

5 Conclusion **15**

1 Introduction

One of the most interesting features of the "Web 2.0" is that it presents a platform where users can easily and freely share their reviews, experiences and stories with others. The increasing amount of such information however rapidly overwhelms the standard user and leads him to loose the overview.

The goal of this project was to implement a city-map, in which travel-related information, such as hotels, restaurants, shop and sights, are directly displayed, and thus give a quick overview of all the relevant locations. Moreover the markers (icons) on the map also show how this location was rated by other users.

Figure 1: Icons. In the first row Hotels rated from 1 (red, bad) to 5 (green, good). In the second row icons for restaurant, shops and sights respectively

As an example, if a user searches for an hotel on a traditional website with travel information, he gets a simple list of hotels. Maybe this site even has a map which shows numbered pins corresponding to the position of the hotel in the list (see figure 2), but he has no indication of sights or restaurants in the neighbourhood and has always to refer to the list if he wants to get the details of an hotel or the rating.

In the implementation of this project, the user sees at first glance how an item was rated. With a single click on the icon he can read a description of the hotel and in a second tab the reviews and experiences of other users.

A third tab gives the user even the possibility to refine his search based on the currently selected item. This example-critiquing makes the map more interactive and guides the user in formulating his search criteria [11, 10].

In the following, I will first present other websites offering interactive travel maps, then present the interface of my map and continue with the implementation details.

2 Related work

I essentially explored three major travel websites: TripAdvisor (www.tripadvisor.com), Yahoo! Travel (travel.yahoo.com) and Home&Abroad (www.homeandabroad.com), all of them providing interactive maps. Their interactivity, however is very limited. A user only can click on an item on the map to view a description.

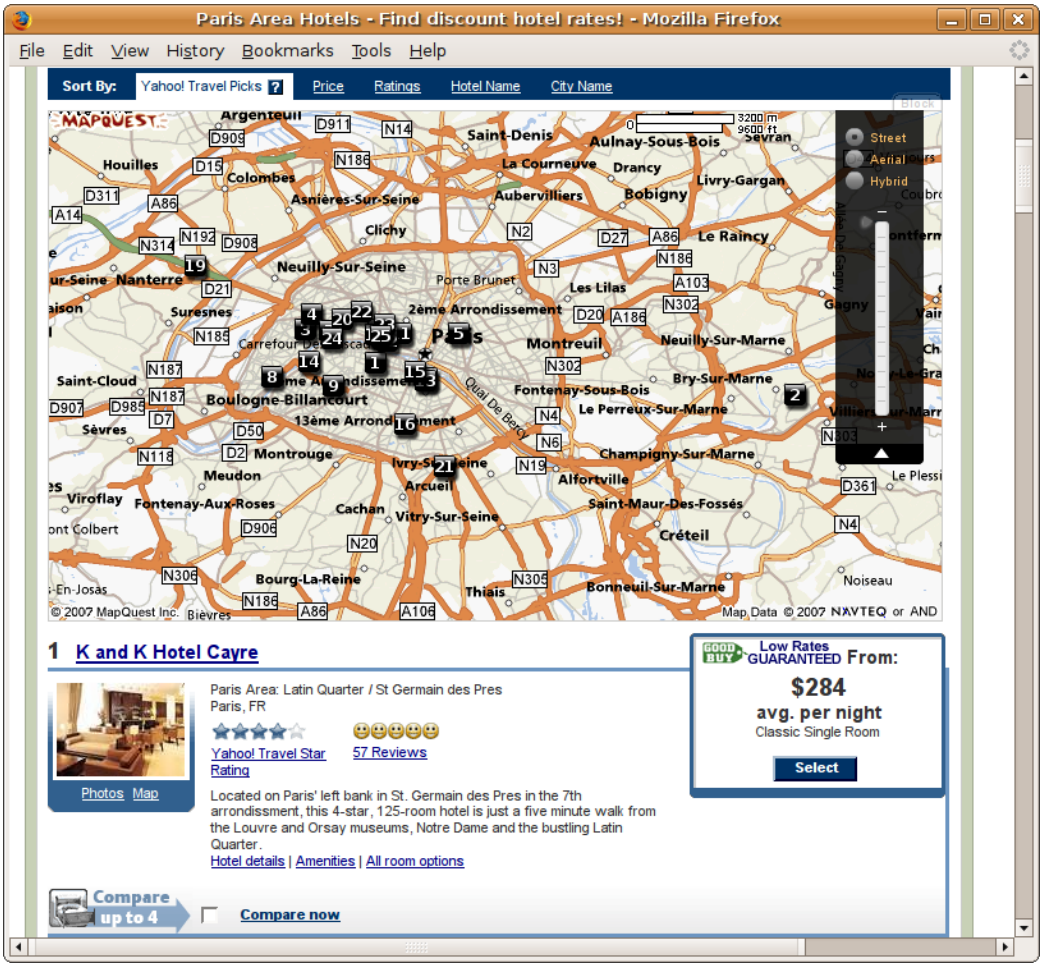


Figure 2: The Yahoo! Travel interface presenting a list of hotels and the corresponding pointers on the map

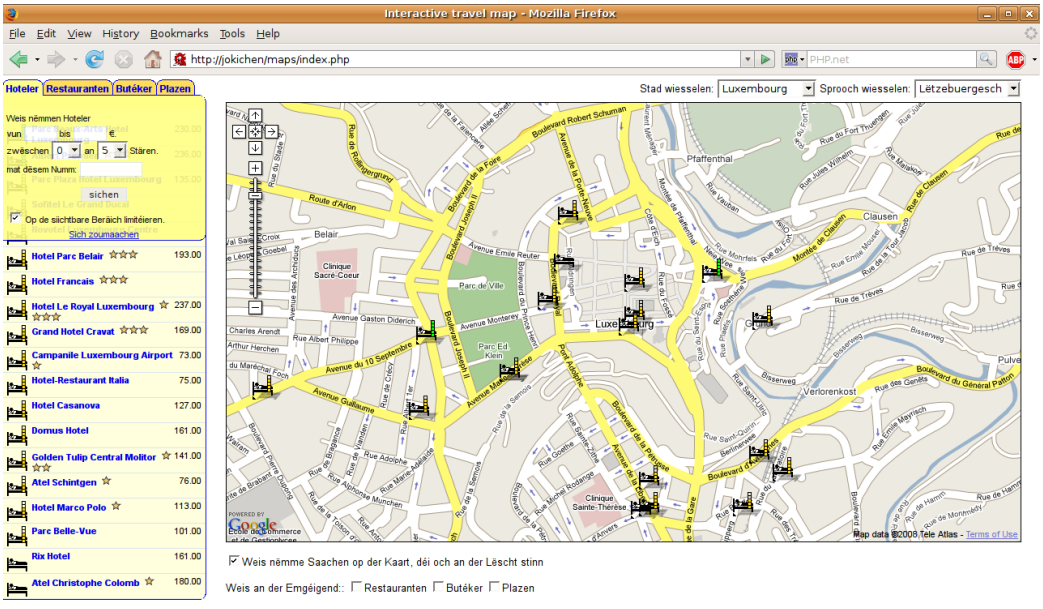


Figure 3: Interface of the interactive travel map

Figure 4: Dropdown box to choose an other city, grouped by country

In order to view user reviews, he is redirected to a new page which then shows the reviews. Home&Abroad and the newest version of Yahoo! also provide the possibility to show other items nearby.

None of these three sites however allow an easy filtering of the list and not at all example-critiquing features.

3 Interface

The interface is basically divided in two parts. On the left side a list with items and on the right the map. In the top right corner are two dropdown boxes to let the user change the language and city. A change of one of these boxes will cause the reload of the page with a GET parameter corresponding to the selection. This parameter will then be registered in the session.

3.1 The list

The list is presented with four tabs, one per travel item type. The selection of a tab will cause an update of the map with the items selected.

Just underneath the tabs is an area for a search section. This part has a light transparency and lies over the list. The search section can be closed by clicking on "close search". An alternative would have been to place this search box above the list, but I preferred the first option, as it requires less space and thus fits better on smaller screens. The search criteria available

Figure 5: The list of items.
Here with hotels, containing "eiffel" in their name, with a price between 50 and 220€.

for the moment are by price and category, if this information is available, and by name, or a part of it. A click on the search button updates the list and the map according to the specified criteria.

There is also a checkbox named "Limit items on the list to the visible area", which is checked by default. This causes the search to be limited to the zone that can be seen on the map.

The number of items in the list itself is determined by the height of the window, in such a way that it fills up the window, but doesn't let scroll-bars appear. Two links, "<previous" and "next>" let the user to switch to the previous, respectively next page.

A list item is composed of an icon (the same as on the map) corresponding to the rate obtained, followed by its name and, for the hotels, the number of stars. Hotel and restaurant items also contain the average price on the right side. If the user moves the mouse over an item, it is highlighted in the list, but also the arrow of the corresponding marker on the map becomes red. A click on an item has the same effect as a click on the marker on the map.

3.2 The map

The map is initially centred to the city centre, or rather the point that Googles geocoding service returns as centre. It has the standard controls known by Google Maps (dragging of the map, scroll wheel to zoom, etc.). A change of the map, like zoom in/out or moving, will cause an update of the items shown on the map.

If the user moves his mouse pointer over an item, the corresponding entry in the list on the left side will be highlighted. The click on a marker opens a bubble, called infoWindow by Google and encircles it, if there exists not yet a reference item (see section 3.3).

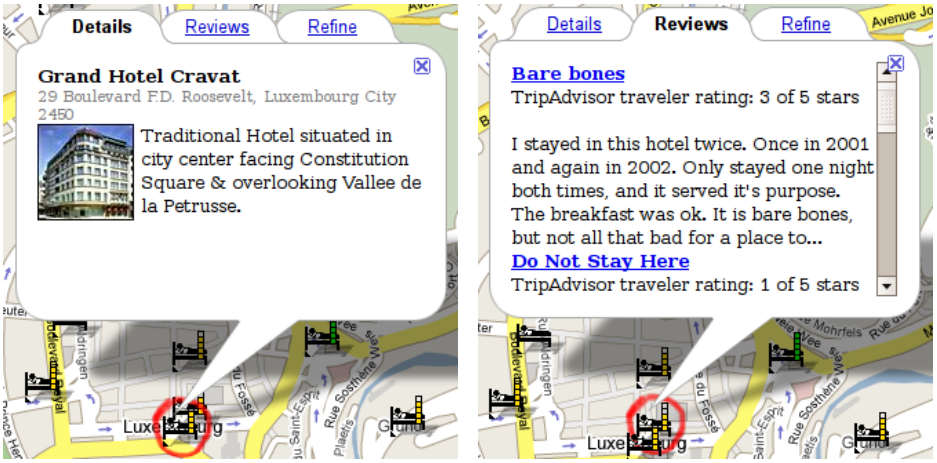
3.3 Infowindows

The infowindow is composed of three tabs: "Details", "Reviews" and "Refine". The content of these tabs is loaded when the window opens. As this may take a couple of seconds, I display a progress indicator in the left corner to show to the user that there is something to come, as suggested in [9]. *(refer to figure 6)*

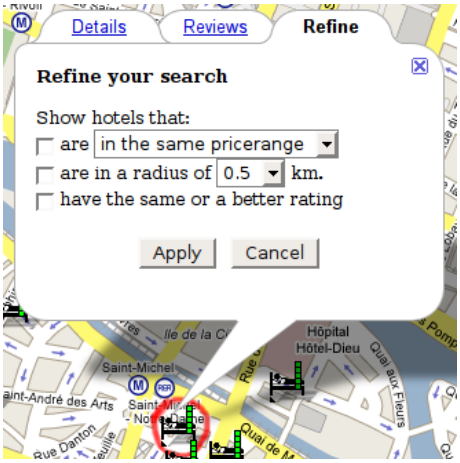
Figure 6: A progress indicator shows that the new information is loading.

The "Details" tab (figure 7(a)) shows the name of the item, the address,

a picture and a description, if available. If there are user reviews, they are displayed in the second tab (figure 7(b)).



(a) details (b) reviews



(c) refine

Figure 7: The three infowindow tabs.

The third tab (figure 7(c)) gives the user the possibility to refine his search based on the selected item. He can for example choose an hotel that is cheaper as the selected one, or limit the results to the neighbourhood of the selected item.

The modification of one of these search criteria immediately causes, in the background, an update of the list and map with the newly specified criteria. If the user clicks apply, these modifications are permanently be applied and the item becomes the reference item and remains encircled. Otherwise, the former state will be restored.

3.4 Other features

Below the map is a checkbox named "Show only items on map that appear in the list". Unchecking this box will cause every item corresponding to the search to be displayed on the map. This may however overload the map and therefore it is checked by default.

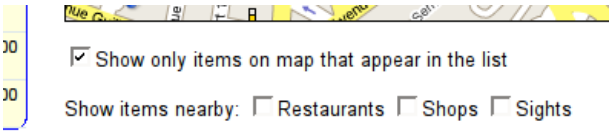


Figure 8: Checkboxes to modify the content of the map.

A line further, there are three checkboxes that permit to display additional items on the map. It might for example be interesting for a user who searches on hotels, to see if there are sights around or if there is a good restaurant in the neighbourhood. In this case he will check "Sights" respectively "Restaurants".

4 Implementation

All data relevant for searches (name, location, rating, price) on the sites is stored in a local MySQL database. The additional data (descriptions, reviews, pictures) is loaded dynamically from the originating site, as soon as the user requests this information.

The implementation can be roughly divided in two parts. A first part are the wrappers, needed to retrieve the data from the source websites. A second, bigger part is the map itself and the interaction with the users.

4.1 Collection of information

In order to populate the database, the required information needs to be retrieved from a webpage. For the moment these are TripAdvisor for the Hotels and Yahoo! for the restaurants, shops and sights, but other sources can be added by simply writing a new wrapper for this specific source.

As none of these source sites allow direct access to their database or provides an api, I need to parse the webpages and extract the relevant data. This is the role of the wrappers: process a page and output all information that we need in a format that can be processed by the application.

The collection of the data takes a very long time, as all the relevant information cannot be found on a single page. Therefore, for every item, I have to parse a separate page, which results in a performance of between 1.5 to 2.5 seconds per item.

4.1.1 Wrappers

The wrappers are written in PHP5 in an object-oriented style. There is one wrapper per type of item and source (e.g. YahooShopWrapper.class.php for the shops from Yahoo!), providing each three public methods:

- *getAll()*: Creates, for each item, an object of the corresponding type of item and passes it to the Register in order to store it in the database.
- *getDetails()*: Parses out, from the page given as argument, all information needed to be displayed in 'Details' InfoWindow and returns them in an associative array.
- *getReviews()*: Parses out, from the page given as argument, all user reviews and returns them in an array.

For all of these methods the document is retrieved from the source site and then treated by PHP's DOM functions [2] with XPath queries and pattern matching to extract all useful data.

An exception to this is the getReviews() method of the TripAdvisor wrapper. Here the SimpleXML functions [3] are enough as the data is already well structured in an RSS-Feed.

4.1.2 Simple form as interface to populate the database

I have written a very simple form that serves as an interface to populate the database *(see figure 9)*. It can be accessed via the file "/getData.php". This allows to specify for which city data will be collected, what type it is, and the part of the source url that identifies the city.

Ideally, in a real-world deployment, the wrapper would be run periodically by a timer.

4.1.3 Database and Registers

The database is kept very simple. It only contains the name, address and rating of an item. The hotels and restaurants also have a category and price field. In addition, every entry contains the geographical coordinates, the links

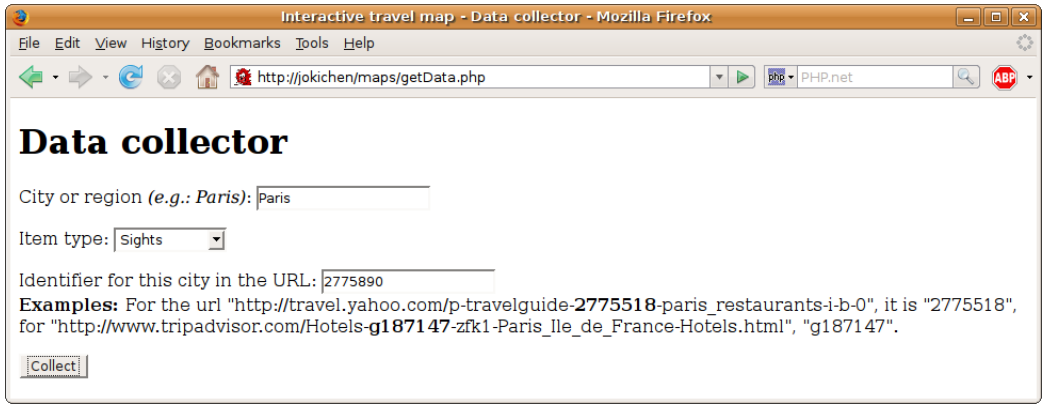


Figure 9: Simple form as interface to the wrappers to collect data.

where the details and reviews can be found and the name of the wrapper to be used.

The registers make abstraction from the database. Every operation that is made on the database goes through a register. There must not be any SQL query outside a register. Thanks to this design pattern, a change of the database structure only requires a modification of the register and thus makes the application more adaptable.

4.2 The site class

For every type of travel item, there exists a class, which all inherit from the Site class. Their fields basically contain the same information as a corresponding tuple in the database.

I only want to point out some particularities:

- Addresses: The address of an item is stored as an Address object. If the constructor of this Address object doesn't get the geographical coordinates as argument (e.g.: a new item has to be inserted in the database), it calls the Google geocode service to get these coordinates.
- Prices: The prices may not always be in the same currency. Therefore prices are stored in a Currency object which should to this conversion in a future version.
- Rating: Ratings are not always given in the same scale. They are therefore stored in a scale between 0 and 1, and can then be retrieved on a user defined scale (here from 0 to 5).

4.3 Setup of the map

As map, I use the Google Maps, with the corresponding api [4] in version 2.92.

The document output by php only contains a place holder for the map. The map itself is loaded with JavaScript by specifying an "onload()"-function.

This function initializes the map. It first determines the size of the browserwindow and then resizes the map to use the maximal available space, but in such way that all fits on one screen, so that the user does not need to scroll. This resizing is also done every time the size of the browserwindow changes.

After this, it places the controllers for zoom and navigation, loads the icons for the markers that may be placed on the map and specifies their properties. These properties include the icons to use, an icon for the shadow on the map, the dimensions of these two icons as well as the edge that will point to the map and the position where the infowindow should point at. Further, this function sets the eventlisteners to the map, amongst others, the reload of new items if the map is moved.

Once the initialisation done, the map is centred to the chosen city (Paris, by default) and then the list of hotels is requested from the server.

4.4 Interaction with the server

The only case where the page is entirely reloaded, is when the city or the language changes. In all other cases, the interaction with the server is done trough XMLHttpRequests, a technology that is also known under the name of AJAX(Asynchronous JavaScript and XML). Despite of the XML in the name, it turned out that using JSON (JavaScript Object Notation) is much more convenient as XML, because it can be directly used as a JavaScript array and PHP has a built-in function that directly returns the JSON representation of a PHP-variable. Although a JSON message can be parsed by the eval() function of JavaScript, I use an external file of functions that facilitate the work with JSON [1].

In order to facilitate the Ajax-requests, I have written a function AjaxRequest() to handle these requests. First it creates an XmlHttpRequest object using Googles GXmlHttp factory method to avoid having to write the code for a normal browser and again for every special case of Internet Explorer. It also handles errors and displays a progress indicator until the requested data is received. Finally it calls the callback function passed as parameter.

An AjaxRequest is generally sent to the /list.php file with as argument the type of item (hotel, restaurant, shop or sight) and the method to call on

Figure 10: Progressbar displayed while data is being retrieved through an AjaxRequest()

the corresponding List Object (getItems(), getDetails(), getReviews() and getRefinementOptions()).

4.5 Getting a list of items

To retrieve a list of items, an AjaxRequest calling the getItems() method is issued, with as callback function display(), which will be described below. But before, the getItems() method will check if any search criteria was passed as argument and if applicable the search criteria are saved in a session variable to memorise them for further refinement queries.

4.6 Displaying the items

For every item returned by a request, a GMarker is created and stored in a JavaScript array. The first n items are then displayed on the list and, if the checkbox "Show only items on map that appear in the list" is checked, these n markers are set on the map, otherwise all the markers are set.

There might however already be markers on the map. It would be to time-consuming to every time remove all markers and replace them. Therefore I have to keep track of markers already on the map and only add the new ones, respectively remove the unused. A big help in the handling of the markers is PdMarker [7], which in particular extends the GMarkers with the ability to be referenced by an id.

But the placement of the markers is still quite time-consuming, and as JavaScript does not know the notion of threads, the user gets the impression that the browser is not responding. To avoid this I had to split up this display procedure and call the setting and deletion of every single marker with a *setTimeout()*. This causes an interrupt after every operation and thus gives back to the user the feeling of reactivity.

4.7 Getting the details and user reviews

The content of the details and reviews tab is retrieved via an XMLHttpRequest. As the description and user reviews are not stored in the database and in order to be able to always display the newest information, this data is taken directly from the originating webpage through the wrappers *getDetails()* and *getReviews()* methods.

4.8 Localisation

As all text is saved in two separate language files (one for JavaScript and one for PHP), it is very easy to translate the application in an other language. It is sufficient to add two new language files and name them *languagecode*.js respectively *languagecode*.php.

4.9 Special treatment for Internet Explorer and issues

As already mentioned above, Internet Explorer (IE) uses 2 different ActiveX objects for XMLHttp instead of JavaScript. To avoid problems, I use Googles factory method.

A second problem I encountered, is that IE does not support the standard way to determine the size of the browserwindow. I found 2 functions that solve this problem at [5].

An other, minor problem is that IE does not yet support transparency.

All features of the interactive map should however work on a normal, standard-compliant browser (based on Gecko or KHTML).

The system of grabbing the content is also very fragile. If a source page changes its HTML-structure, the loading of the elements using this source will fail until the corresponding wrapper is updated. This recently happened with Tripadvisor, forcing me to completely rewrite the wrapper.

5 Conclusion

This project shows a way to display travel related information on a map and integrate the benefits of "Web 2.0" (in form of user reviews) and example critiquing. It presents a very simple but efficient interface that does not overwhelm the user with information.

For me, this project was a good opportunity to learn the benefits of AJAX and how to use a Google API (in particular Google Maps). It permitted me also deepen my knowledge of Javascript. A good help for me were [8] and first of all [6].

I also want to point out that the webpage passes the W3C validation [12] and is valid XHTML 1.0 Strict.

List of Figures

1 Icons. In the first row Hotels rated from 1 (red, bad) to 5 (green, good). In the second row icons for restaurant, shops and sights respectively 4
2 The Yahoo! Travel interface presenting a list of hotels and the corresponding pointers on the map 5
3 Interface of the interactive travel map 6
4 Dropdown box to choose an other city, grouped by country . . 6
5 The list of items.
Here with hotels, containing “eiffel” in their name, with a price between 50 and 220€. 7
6 A progress indicator shows that the new information is loading. 8
7 The three infowindow tabs. 9
8 Checkboxes to modify the content of the map. 10
9 Simple form as interface to the wrappers to collect data. . . . 12
10 Progressbar displayed while data is being retrieved through an AjaxRequest() 14

References

[1] Json in javascript. *http://www.json.org/js.html*.

[2] Php: Dom. *http://www.php.net/DOM*.

[3] Php: Simplexml. *http://www.php.net/SimpleXML*.

[4] Google maps api. *http://code.google.com/apis/maps/*, 2008.

[5] Der-Albert.com. Fensterhöhe und fensterbreite im internet explorer und co. *http://der-albert.com/archive/2006/03/21/fensterhoehe-und-fensterbreite-im-internet-explorer-und-co.aspx*, 2006.

[6] Danny Goodman. Javascript & dhtml cookbook. *O'Reilly*, 2007.

[7] Peter Jones. Google maps extension: Pdmarker. *Pixel Development, http://www.pixeldevelopment.com/pdmarker.asp*.

[8] Michael Mahemoff. Design patterns en ajax. *O'Reilly*, 2007.

[9] Jakob Nielsen. Response times: The three important limits. *Usability Engineering*, 1994.

[10] Pearl Pu and Li Chen. Integrating tradeoff support in product search tools for e-commerce sites. *Proceeding of the ACM Conference on Electronic Commerce (EC'05), Vancouver, Canada*, pages 269–287, 2005.

[11] Pearl Pu and Pratyush Kumar. Evaluating example-based search tools. *Proceeding of the ACM Conference on Electronic Commerce (EC'04)*, pages 208–217, 2004.

[12] W3C. Markup validation service. *http://validator.w3.org/*.