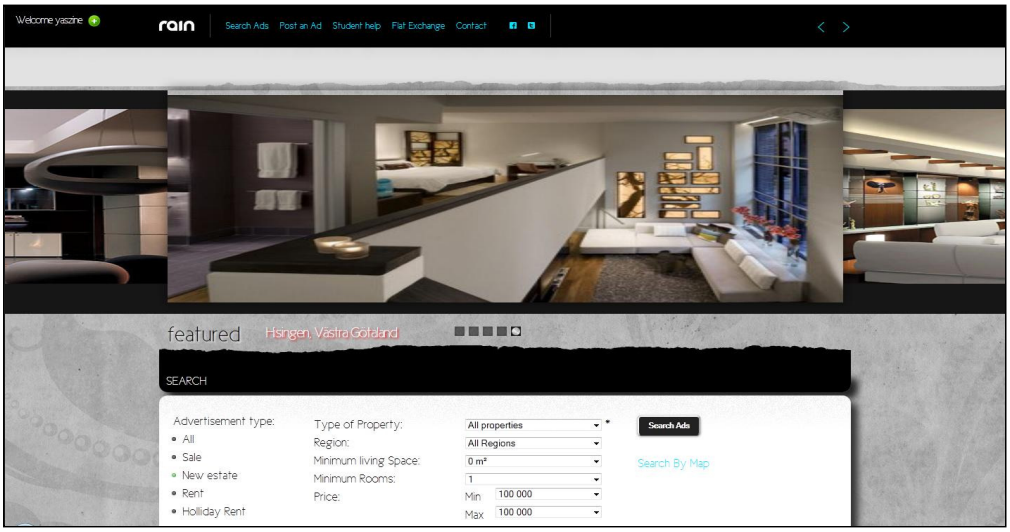
CHALMERS



An international Real Estate Advertisement System

Advertising Real Estate world-wide with European Business Partners HB

Master of Science Thesis in Computer Science.

YASSINE LAMRANI ABOU ELASSAD

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
Göteborg, Sweden, June 20111

AN INTERNATIONAL REAL ESTATE ADVERTISEMENT SYSTEM

Advertising Real Estate world-wide with European Business Partners HB

YASSINE LAMRANI ABOU ELASSAD

Examiner: Dr. Jan Skansholm

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
Göteborg, Sweden June 2011

Preface

This project is done under the supervision of European Business Partners HB. It is one of the many motivational projects the company tries to achieve to further root itself in the market. This report aims to discuss the idea behind the project, the implementation phase, and the results.

This thesis is the last part of my Master of Science degree at Chalmers University of Technology, Sweden. And I would like to thank my patient and helpful examiner, Professor, Dr. Jan Skansholm, and my two sponsors and supervisors at European Business Partners Sebastian and Stefan.

Abstract:

There are over **324,697,205** [4] websites on the internet as of May 2011. What I'm aiming to create, conjointly with European Business Partners HB, is not just yet any other website or electronic service. Indeed, the goal is to create an international space where users from across many countries can advertise a property or search for one in the nicest and easiest way possible. By making the website international, the average customer wouldn't have to try and look for the real estate website local to every country.

Take for example the case of an American student wanting to go to France for studies. Normally he'd try Google to search for a website that advertises French apartments. The search word he types would usually be in English. However a good local website will most likely be in the country's language, which in this case is French. And so, the student in our example would have difficulties progressing with his travel.

Consequently, the project's purpose is to provide a solution to this type of problem and many others. In this report, then, I will be discussing some of these issues related to the world of real estate advertisement, some improvements I thought of, and the results I have achieved.

Some of the results that illustrate a solution to the aforementioned problem would be to add a translation tool to the website. So that even if a property's description is written in a different language, it would require little effort to translate. Also, the general solution I am proposing is to aggregate all ads of every country inside one website. And to do that, two main steps are necessary:

- First: Build an attractive website that offers as many facilities and innovative features to the user as possible.
- Second: Market the website, and publicize it.

To achieve the second step, the company and I decided to first start by marketing the website in one country at a time. First, we'd start by a general English website, which would serve as our template website. Then this latter one will be translated into Swedish, and it will be marketed in Sweden. Indeed, the approach is to market it in one country at a time, until it gains strong popularity. And for each of these countries, the website will provide geographical, language, and technical support.

Overall, throughout this report, I will provide more in-depth explanation of the problem at hand, a thorough investigation of possible solutions, and finally the results achieved, with examples of the implemented website. I will also discuss at the end all future work still underway.

Table of Content:

1. Introduction

A website's worth in today's market is highly dependent on the number of its users and daily visitors. A great business investment would be then to realize such website that would attract the highest number of customers and crush its competition. By doing so, the site's revenue could ever more keep on growing, through its popularity, leading to the creation of its own trademark and expansion of the organization's profit. Following this very same logic, the project I have undertaken aims to create a real estate advertisement website that would appeal better than the existing ones and attract more customers throughout time.

It is however, hardly evident on how to achieve such goal. There already exists many websites for real estate advertisement. Indeed, the main competition we face is no push-over:

1. ***Blocket.se:*** a multipurpose advertisement website with over 3700 ads for real estate properties and a net worth of over 1.8 million US dollars.
2. ***Hemnet.se:*** Specializing mainly in large estates, this website has over 51000 ads.

These websites have existed far earlier than the one we are implementing. And as a result, the main problem and challenge we face is that the Swedish market is already accustomed to these existing websites. It would require a lot of continuous work and original ideas to replace these with one that would deserve even higher numbers, both in customer count and worth.

Consequently, the approach we adapted was to add a modern interface to our system with an enhanced and more appealing design, along with better, and most needed innovative features than the existing ones. Nonetheless, we tried to keep the overall feel and use of the website as simple as what the users are already accustomed to. As a matter of fact, too much of a new experience could simply achieve the opposite of what we want, and instead of attracting users, it could drive them away, to their old ways.

So the main questions we tried to answer and implement were:

1. How to attract the largest possible number of customers?
2. What could we do, that others don't have?
3. How to make the user experience more enjoyable?

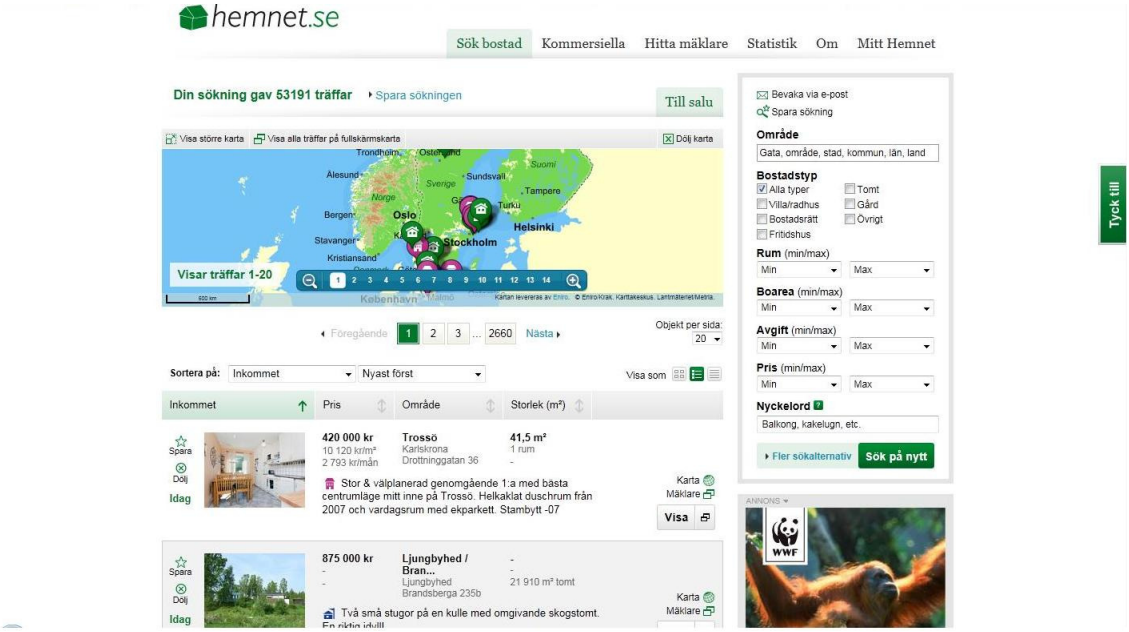
In the following report, I will investigate different solutions and answers to the aforementioned questions, discuss the results, and define the outcome of these investigations. I will also, later, describe the practical implementation of these results in our website.

The website will support the basic functionalities of posting an advertisement, searching through the advertised properties, and displaying results in the most intuitive manner, along with the extra features we decided to add.

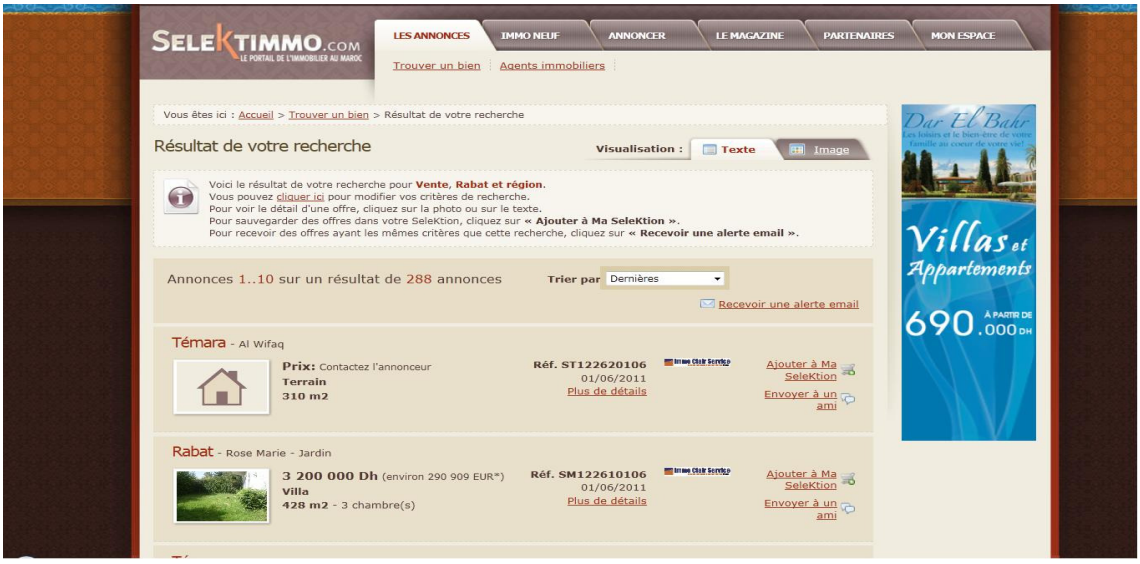
2. Background

As it is said in the introduction, our main competitors in Sweden are Hemnet.se and blocket.se. But if we mean to make the website international, we need to consider any online real estate website as our competitor. For this reason, we investigated websites with the most users, to have an idea what the real estate web market is like. In the following, I list the link and country to some of such websites from each country, along with a snapshot of their result listing since it's the most important in our work.

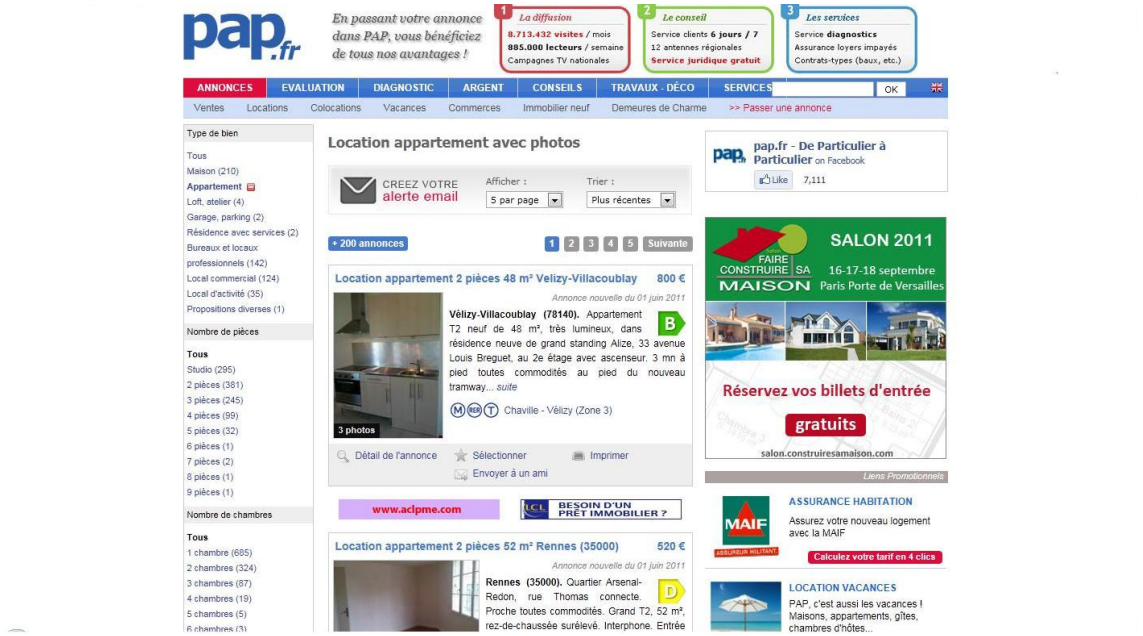
- ***http://Hemnet.se - Sweden:***



- ***http://www.selektimmo.com/ - Morocco:***



- ***http://www.pap.fr/ - France:***



Of course these websites are not the best ones in terms of design, but nonetheless they are successfully used by many for the features they provide. And thus our aim is to offer at least the same functionality and features they have.

3. Literature review

The language we picked for our project is Microsoft Active Server Pages or ASP along with the .net framework (or in short, asp.net). Asp is a standard created by Microsoft in 1996 with the aim to develop interactive web applications, as in applications with dynamic content. [5]

Actually, ASP.net is a technology or more precisely, a programming environment, rathen than a programming language. It allows to present, in the form of objects, the interactions between the client's browser and the web server, as well as the connections to databases (thanks to ADO, ActiveX Data Objects) or COM(Component Object Model) components. ASP pages are therefore run at the server side (same as CGI or PHP scripts...) and not at the client side.

First, before we go on to further discuss the reasons behind choosing ASP as our programming language for server side scripting, let us first investigate what other choices there are.

Indeed, when it comes to server side coding, a programmer will be faced with a wide variety of possibilities. Some examples of such languages are as follows [14]:

- ASP.NET
- ASP
- JSF
- CSP
- ColdFusion
- CGI/Perl
- Groovy/Grails
- Java
- Lotus Domino
- PHP
- Python
- Real Studio Web Edition
- Ruby
- Smalltalk
- SSJS Server-Side JavaScript
- Websphere

3.1 Main Server Side Programming languages

Trying to narrow our selection of choices will leave us with the following 4 other prominent server side programming languages in the world of web developement [6]:

- **CGI/Perl**: being older than the world wide web itself, it constitutes a mature cross plateform language not specifically created for web developement. It is a multipurpose scripting language, that can do more than just web developement, designed for large text manipulation from many source,. And since this is mostly what server-side programming evolves around, Perl became one of the strong server

side scripting languages. Perl functionality can be, and usually is, expanded through means of modules. These modules help achieve many specialized purposes, such as creating dynamic web content. This is done through the Common Gateway Interface (CGI) standard that allows transmitting the client's request to any program of choice. The only downside to this language is that, since it wasn't purpose-built for Web development, it hasn't been optimized either for speed, scalability, or ease of use in a Web server setting.

- **Cold Fusion:** a powerful easy web development programming language which uses tags for programming, as in HTML and XML, instead of scripting. The main strength of this language is the ease and short time in which a non-experienced programmer can build interactive webpages. Indeed, it allows developers to build complex functionality in a short number of code lines. It offers a build-in library of over 300 tags with the option of adding custom tags using C/C++ or Java. ColdFusion is a very costly Adobe product that utlises an Integrated Development Environment(IDE) based on Eclipse.

- **PHP:** or Hypertext Preprocessor is a free scripting language, with familiar simple syntax principally used for dynamic web development. It is an imperative language with complete object models. Because of its rich and comprehensive library, we refer to PHP sometimes as a plate-form rather than a simple language. It's Open Source, and integrates with all main Web servers on all main operating systems.

- **JAVA:** Java Web Development is based on small Java Programs (Servlets) that hands over the client requests to JavaServer Pages (JSP). This latter one is a technique which allows developers to generate dynamic xml or html code along with any other type of web pages. This technique allows the Java code and some other predefined actions to be added inside a static context. The JSP syntax is based completely on xml in form of tags called JSP actions. These tags can then be used for function calls. The tags libraries offer an independent method for the plate form to extend the functionalities of an HTTP server. JSPs are then compiled to become a Java servlet. In fact, a JSP compiler can generate a java servlet in Java source code, which can in turn be compiled by a Java compiler or generate the pseudo code Java directly interpretable. Overall, the main advantage with Java server coding, is that it uses a familiar language with the paradigm "write once, run anywhere" which makes it very powerful.

- **JSF:** Java Server Faces, a java framework, for web development. It is part of the Java EE standard and is the fruit of JCP (Java Community Process). In the contrary to the other traditional MVC frameworks based on action, JSF is based on the notion of components, similar to those of Swing, or SWT, where the state of a component is registered during the realization of the page, before being restored at the request

return. JSF uses JSP by default, but can also be used with other technologies such as Facelets, XUL, or Servlets. More precisely, it Benefits from the use of Struts along with other JAVA technologies such as Swing with its event-driven model. It allows developing client-rich interfaces all the while respecting the MVC paradigm [13].

3.2 Comparison of the potential Server-side Programing languages

In the following chart, I'll be summarizing the main advantages and drawbacks of each of the languages described above, as part of the choice process. [6]

Language	Advantages and Drawbacks	
CGI/Perl	Advantages	▪ Free ▪ Robust ▪ Supported by most web hosts ▪ A large network of users ▪ CPAN archive with many examples
	Drawbacks	▪ Doesn't scale well especially on busy servers. ▪ Has many ways to do the same thing which makes it hard to cross use different examples ▪ Weak performance on a windows server ▪ No formal support ▪ Not meant for Web development implies that it's not optimized for Speed, Scalability, and ease of use in a web server.
ColdFusion	Advantages	▪ Extremely easy to use ▪ Scalable ▪ Adobe's professional support ▪ Cross-platform
	Drawbacks	▪ Very expensive ▪ Not really meant for experienced programmers
PHP	Advantages	▪ Free and Cross Platform ▪ Easy to learn ▪ A large user community ▪ Costs nothing extra for hosting ▪ Relatively Fast
	Drawbacks	▪ Slightly unprofessional ▪ Open source with no official help
JAVA	Advantages	▪ Write once, run anywhere ▪ Powerful and Scalable ▪ Mostly free for personal use and development. ▪ Cross platform ▪ Separate presentation from content through JSP

	Drawbacks	▪ Java plugins need to be paid for if used to host commercial website. ▪ Not straightforward to use for Web development
ASP.NET	Advantages	▪ Option to use C# as a server-side scripting language makes for an easy development experience ▪ Offers powerful dynamic Web development capability ▪ Scalable and easy to secure ▪ Professional support and large examples library.
	Drawbacks	▪ It costs extra to host on a web server ▪ Relatively slow since it runs mostly on windows which is accepted to be slower than linux.

In the above chart, I omitted discussing the specific technical strengths and weaknesses of each language in detail. They all mostly provide almost the same functionalities and utilities such as easy data binding, page templating, enhanced security, and so on. As a matter of fact, comparing all the technical aspects of each language is a whole research area on its own. So at the end, deciding which one is the best remains much more a personal choice than a technical one.

3.3 Asp.net: the server-side programming language of choice

3.3.1 Reasons of choice

Besides the obvious advantages stated above, we picked ASP as our programming technology, along with C# as the server side programming language of choice, for the following reasons [1]:

1. ASP is a professional environement used by most companies to develop professional robust software.
2. ASP's hosting cost has little significance since the website is sponsored by the company
3. Using ASP constitutes an important learning experience.
4. It is possible to further develop specific web applications for the website using SilverLight with the same learned material.
5. Given the short time frame for the Website's development, ASP's time to learn fits perfectly.
6. It is expendable and complete.
7. C# is a powerful object oriented programming language that offers many functionalities.
8. It is possible to call javascript functions through C#
9. ASP supports adding many features to our website (such as google maps)
10. In all modern browsers (Firefox 4, chrome, IE9,..), ASP runs seemingly just as fast as any other server-side technology.

3.3.2 Technical reasons

ASP.NET's advantages are many. I site a few of them in the following [2]:

- It offers Better language support
- Provides Programmable controls, and has a rich toolbox.
- Supports Event-driven programming
- Uses XML-based components
- Has a Built-in User authentication, with accounts and roles
- Offers Higher scalability and Increased performance through Compiled code.
- Enables Easier configuration and deployment and building large web applications requires significantly less code
- It is a purely server Sided Technology and makes for easy deployment since it uses a dedicated file for built-in configuration

3.3.3 ASP Description

ASP pages are embedded inside an html web page through the use of special tags. These tags allow the web server to know that the code within has to be executed before sending back the data to the client's browser. In this way, the Active Server Pages constitute a 3-tier architecture. This means that a server supporting ASP can serve as an intermediary between the client's browser and a database allowing for a transparent access to this latter one thanks to the ADO(ActiveX Data Objects). ADO is responsible for providing the necessary elements to connect to the database management system through the use of SQL statements. [5]

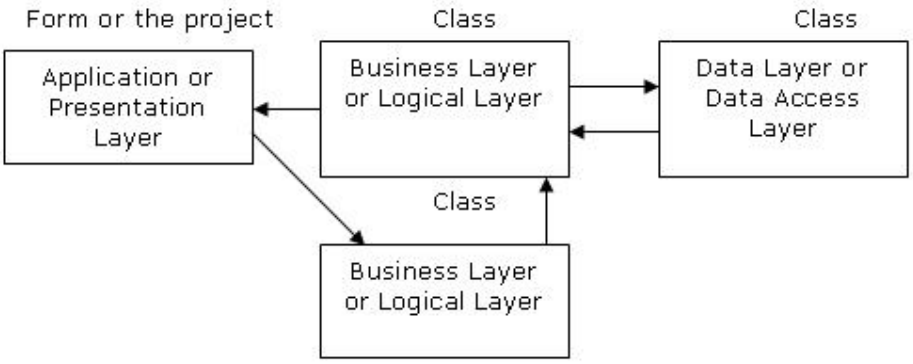


Figure 1: Figure illustrating the 3-tier Architecture nature of ASP (ie. The interaction between the client's browser, the Web server, and the database)

3.3.4 ASP's characteristics

ASP pages were conceived initially to function over Microsoft Web server IIS (Internet Information Server). This web server, created by Microsoft in 1996, is a free server, functioning on top of the Microsoft Windows NT operating system. Now, however, this proprietary technology is available over web servers other than Microsoft's IIS. First it was ported over Netscape's FastTrack server by Chili Software, and then it was ported over other

servers such as Apache, through the Apache::ASP module. Consequently, this module makes it possible to create website utilizing the ASP technology over numerous various platforms (such as, Unix, Linux, PowerPC, …).

ASP pages are based on objects manipulated by the server permitting to realize many applications. The 7 basic objects are as follows:

- Application: An object representing the Web application itself. The object contains all the shared information between visitors connected to the application (e.g. number of users connected simultaneously,…)
- Object Context: Allows controlling the potential transactions with the Microsoft Transaction Server (MTS).
- Request: this object is used to get information sent to the server in the client's http request.
- Server: contains the web-server relate information
- Session: allows managing a user's session (i.e. conserve the user's information from one page to the other)
- ASP error: It catches any errors, and defines them, that happen at ASP scripts execution. [5]

3.3.5 ASP Programming

An ASP script is a simple text file containing instructions written in ASCII 7 bits, existing on the server with the extension .asp or .aspx. To allow the server to decide whether the code should be interpreted or simply resent as is to the client, specific tags are used. It is then possible to mix html and ASP code inside the same file. The ASP scripts are then text files in which we find ASP instructions along with the response text (html, xml, or JavaScript code).

Therefore, when a client wants to access an asp page:

- The server recognizes that it's dealing with an asp file thanks to its .asp, or .aspx extension.
- It reads the file
- When the server encounters a tag indicating that the following lines are ASP code, it executes them
- When the server encounters and instruction, it transmits it to an interpreter.
- The interpreter executes the instruction then sends back any results to the server
- At the end of the script, the server transmits the results to the client.

ASP scripts are interpreted by the server, so the users cannot see the source code. It uses tags format to define which scripting language used as follows:

```
<SCRIPT language="VBScript|Jscript"
                [runat="server|client"]
                [type="text/vbscript | jscript"]
                [src="url"]>

Script Code
</SCRIPT>
```

Also, ASP provides an alternative way to separate the codefile from html file via the following tag:

```
<%@ Page Title="" Language="C#" MasterPageFile="~/Site.master"
AutoEventWireup="true" CodeFile="mypage.aspx.cs" Inherits="Result" %>
```

The codeFile contains all the server side code written in the language of choice (either c# or VisualBasic) responsible for data manipulation. [5]

4. Scope and End Goal

4.1 Goal of the project

The goal of the project is to create a real estate advertisement website that stands out of the existing ones. In today's competitive world of Web Interactions, it is important to have a significant online presence that attracts the maximum number of loyal customers. Therefore, it is imperative to create a brand for the website. The brand will constitute the bridge to connect with the clients. More precisely, the aim is to convey to the market that we are the ones the best ones to provide them with a solution to their real estate search or advertisement.

The most important part of the project is then to investigate and understand what any client, like me and you, wants and needs. Initially, a customer is looking for a property in a specific area either for rent or sale. When he finds the websites offering such service, he then picks the one that presents itself to be the most easy to use, with the most promising features and most attractive design.

The most attractive the website is, the more likely the client will trust in it. It has to be easy to use for the average customer, so that he'd keep on using it throughout time. In short, it has to have the feel and look that it has everything a consumer needs.

In this respect, technically, the website needs to have the following aspects:

1. An attractive design
2. Features allowing for the ease of use
3. Exhaustive information easy to access
4. Interactivity with the user.
5. A modern feel and look.
6. Possibility to expand and scale easily

More specifically, from a programmer's point of view, the Real Estate Website features should include:

a. A comprehensive search mechanism
b. An attractive Slide show for the property
c. Fully editable user pages
d. A Featured properties section
e. Capability to be found by Search Engines easily
f. Fast Search Results
g. Interactive Map functionality
h. Detailed advanced Searches
i. Email capability
j. Personalized user space to save searches and favorites.
k. Sorting capabilities for results
l. Ease of access and browsing

4.2 Limitations

4.2.1 Technical Limitations

Web programming, and website development is a multi-tasked engineering field. Web applications now comprise of multifaceted functionality with complex behavior. Developing a dynamic fully-fledged robust website then requires high demands on its usability, performance, security and scalability. More precisely, web engineering uses software engineering principles, along with new approaches, tools and techniques to satisfy the requisites for successful web development. The developer needs to be able to address issues of maintainability, quality, and reliability. [9]

Therefore, for a lone programmer, this project can be very demanding and extremly time consuming. Actually, I work alone on implementing this website, while being highly unfamiliar with web development. Therefore, as part of my thesis work, I was required to learn and practice the aforementioned skills along with the programming language of choice. This, combined with information gathering, constituted an important and highly valuable part of the project, and made for an experience of the utmost significance that would last a lifetime.

4.2.2 Miscelleneous limitations

A few other limitations to the project realization are as follows:

- To realize an international website with various languages, we need experienced translators or native/fluent speakers of the target language.
- We have limited knowledge of the Real Estate world.

4.3 Scope

Given the limitations stated above, we should differentiate between the scope of the thesis work, and the scope of the project itself.

The scope of the thesis work is mostly the same as the scope for the project within half its original time frame. It encompasses the following:

1. Investigation of Programming languages to choose
2. Learning the essential languages
3. Setting up the programming environment and getting used to it
4. Information Gathering, and discussion of project ideas.
5. Website implementation sampling
6. A complete website implementation in English with a beta design

The scope of the thesis work can be then said to focus primarly on:

- The theory behind implementing a real estate website,
- Implementing the most important features and design ideas
- Database implementation
- Problem investigation and solution making.

On the other hand, since the project is somewhat independent of the thesis itself, we can say that the project's scope is rather very broad. It is meant to be continuously updated and evolving to never fall back behind as many other websites do.

The nature of the project which is based on marketing the website one country at a time, makes it rather very difficult to estimate the completion time. However, the core functionalities and the immediate goal of integrating the website successfully inside Sweden should be finished by mid-july.

Generally, the overall scope of the project, given no time restrictions, is as follows:

- Implementation of core functionalities and extra features
- Include version of the website in a few main major languages
- Test and debug simulation
- Import Real Estate Ads from existing advertisement websites
- Finallize the design or customize for every country
- Improve visibility by google search
- Include payment service after establishment in the market
- Expand to a larger web server
- Improve Security measures

5. Investigation, Discussion and Results

5.1 image display in results

Problem:

When displaying search results in the browser, displaying the images for each property can be done in numerous ways. The basic one is through a miniature size of the first picture in the property's image collection. This is an old method, unfit for a modern website. The problem then is to investigate different ways to show a property's image data within the search results.

Investigation

The investigation led to identifying different possible technics to display the property's images within the search results listing:

- Display miniature pictures (1cm x 1cm) for every property listing
- Use the JQuery library to dynamically change the displayed image, and eventually show all images.
- Use a Flash application to model the images in a very fashionable way
- Use an image gallery implemented either in Javascript or Flash.
- Utilize the power of CSS3 to implement a fashionable display of Images.

CSS3 (or Cascading Style Sheet level 3) offers many new capabilities that render web design much easier. Some of the new features it introduces are:

- Text effects such us shadows and 3D rendering
- Box shadows and round corners
- Custom Web fonts
- Opacity and layers-like effect
- Animation

Solution:

One particular solution for this is through the use of CSS3. The solution consists of displaying all the property's images within the same space. [3]

Within a list of results, the picture part of a property would look as follows:

Hovering over a specific picture (in this case the picture with the title “abstract”) results in it’s exapnsion as follows:

Remark

Both JQuery and Flash offer the same possibility to implement the chosen display of pictures. However using these technologies is relatively less favorable than the chosen solution for the following reasons:

- They require time to learn
- They require a lot of time to implement
- They consume a lot of CPU
- They take time to load and slow down the loading of the website.
- We are looking for an easy solution, and CSS3 requires considerably a lot less lines of code.

On the other hand, the downside with using CSS3 is that it is not possible to compute mathematical expressions to dynamically accommodate the display depending on the number of pictures to display. The only way to do this would be through hard coding different layout styles for different number of pictures to display, then to dynamically choose at the server side which layout style to use depending on the number of pictures the property has.

For example, to reproduce the above menu comprising of 5 pictures, this is a snippet of the code used:

```
<style type="text/css>
	.slidetop {
		width:342px;
		padding:0px 2px 2px 0px;
		margin:0 auto;
		background:#FFF;
		border-bottom:1px solid #000;
		border-right:1px solid #000;
	}
	.slider {
		width:340px;
		height:200px;
		overflow:hidden;
		margin:0 auto;
	}
```

```
      .slider .ulpic
      {
             padding:0;
             margin:0;
             list-style:none;
             width:800px;
             height:200px;
             overflow:hidden;
      }
      .slider .ulpic li {float:left;}
      .slider .ulpic li a
      {
             display:block;
             float:left;
             width:68px;  //width1
             border-right:1px solid #FFF;
             height:200px;
             overflow:hidden;
             -webkit-transition: 0.5s;
             -moz-transition: 0.5s;
             -o-transition: 0.5s;
             transition: 0.5s;
      }
      .slider .ulpic:hover a.p {width:10px;} //width2
      .slider .ulpic a.p:hover {width:300px;}
      .slider .ulpic a#current {width:300px;}
</style>

</body>
<div class="slidetop">
 <div class="slider">
  <ul class="ulpic">
   <li>
    <a class="p" href="#url1">
     <img width="300" height ="200"  src="content_1.jpg" alt="" />
    </a>
   </li>
   <li>
    <a class="p" href="#url1">
     <img width="300" height ="200"  src="content_2.jpg" alt="" />
    </a>
   </li>
   <li>
    <a class="p" href="#url1">
     <img width="300" height ="200"  src=" content_3.jpg" alt="" />
    </a>
   </li>
   <li>
    <a class="p" href="#url1">
     <img width="300" height ="200"  src=" content_4.jpg" alt="" />
    </a>
   </li>
   <li>
    <a class="p" id="current" href="#url1">
     <img width="300" height ="200"  src=" content_5.jpg" alt="" />
    </a>
   </li>
  </ul>
 </div>
</div>
</body>
```

Consequently, this setup is only meant for a set of 5 pictures. We can however easily change and recreate the setup for different number of pictures meant to be displayed.

- For 1 picture: there's no need to use the defined style.
- For 2 pictues: Width1=170px & width2 = 40px
- For 3 pictures: Width1 = 113.5px & width2 = 13.5 px
- …
- For 10 pictures: Width1=33px & width2 = 3px

When processing data results at the server side, we can simply check the number of available pictures for each property and assign the appropriate styling for it.

5.2 Comprehensive Email

Problem

In most cases, the user wishes to send a message to the property owner but doesn't really know what to write about or what information to supply or ask for. On the other hand, the real estate agent is usually looking for specific information from the part of the customer in order to provide him with an adequat reply. For this reason, in the normal case, the agent would be obliged to write back an email to his customer inquiring about this information, before he can provide him with a reply to his inquisition. This, in return, proves to be a time consuming process which most real estate agents wish to avoid. Consequently, A simple email form may prove to be too inconvenient for the owner as it may not supply him with the information he requires.

Investigation & solution

Therefore, instead of using a simple a Email form, I sought to add a few fields to help both the customer and the client communicate with each other.

The investigation then for this problem comprised of conducting a little interview with real estate agents In order to know what relevant fields to add. The main question is what information are they hoping to find when they receive an email from a customer. The result is shown in the following figure:

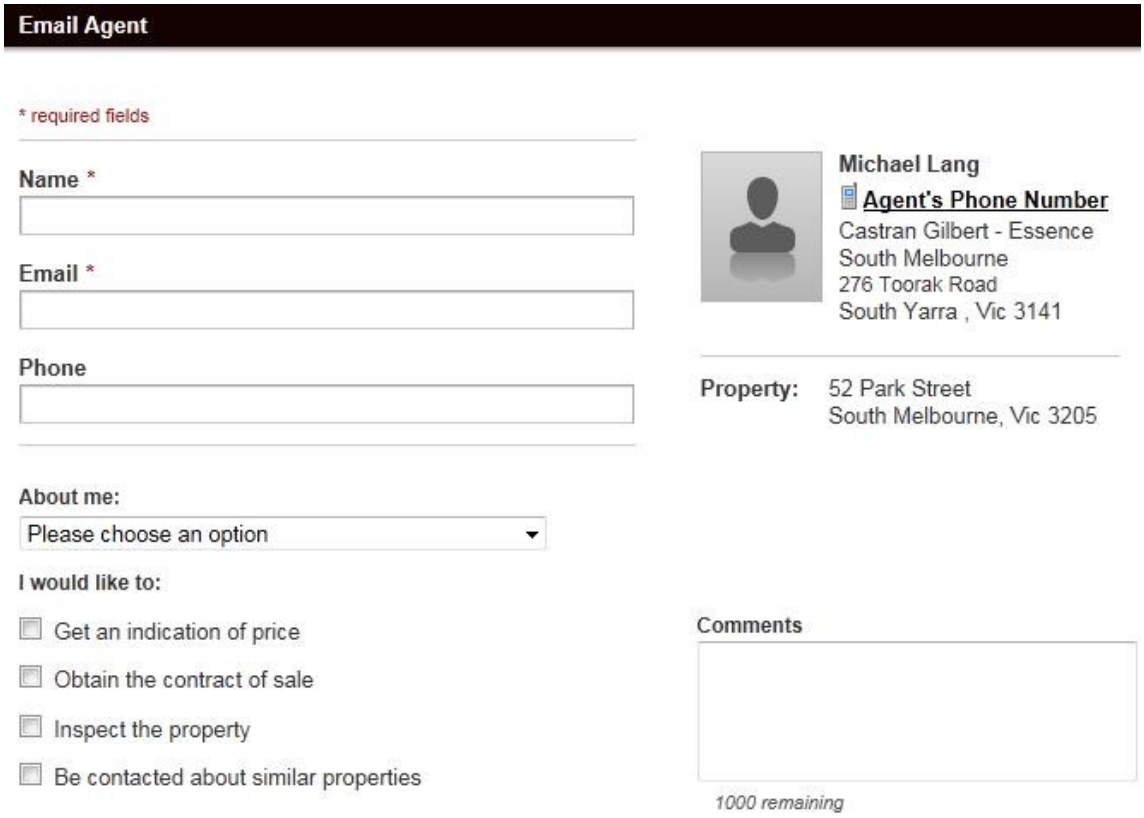


Figure 2: email contact form

The "about me" select field contains the options:

- I own my own home
- I am renting
- I have recently sold
- I am a first home buyer
- I am looking to invest
- Monitoring the market

5.3 Ease of access to information

Problem 1

The main problem in every website in general is that the user is often forced to visit many pages to get the information he's looking for. In the real estate context, in most cases, while browsing through the results page, he would have to visit every property's page in order to get basic related information. Therefore, in this part of the project, I aim to maximize the information a client can know just from the results page without having to go any further. Such information is as follows:

- Property's map view
- Property's images
- Property's description and features

The aim is that the user will be able to know almost everything he needs to know about a property without having to go any further that the results page.

Investigation

The solution to this problem is rather straight forward: pipelining. This allows the user to stay on the same page while accessing other information. The investigation then consisted of finding out the most fashionable and powerful way to achieve this functionality. This resulted in using Colorbox v1.3.17 which is a light-weight, customizable lightbox plugin for jQuery 1.3, 1.4, & 1.5. [11]

We will use the colorbox to display all the images for each property in their original size, and also to display the location of the property on a map.

As for displaying a property's description, it is easier and more efficient to simply use a tooltip that is triggered by hovering over the "description" label.

To use colorbox, we need to initialize its parameters in javascript at the client side, for every set of pictures as follows:

```
<script>
    $(document).ready(function () {
       $("a[rel='set1']").colorbox();
    });
</script>
```

And thus for 50 sets of pictures meant to be displayed at once, it is mandatory to initialize the colorbox parameters for each set.

To display the map, we utilize colorbox's ability to embed an iframe. And we implement an external web page containing the map utility. The result looks as follows:

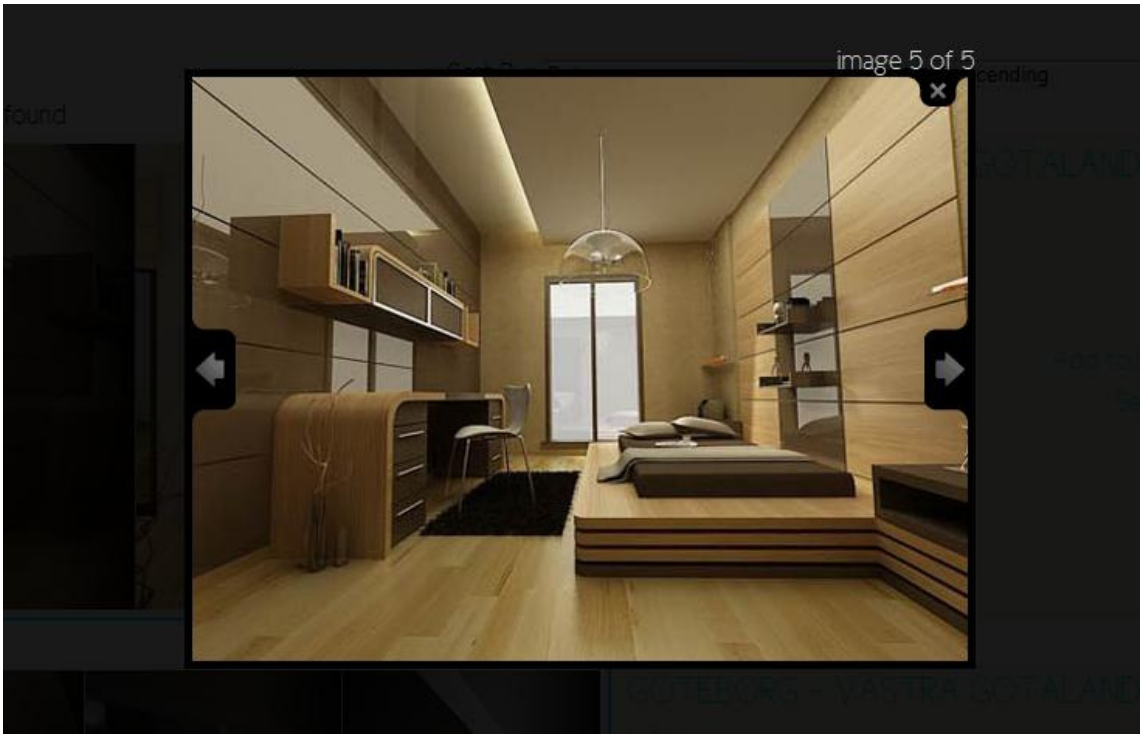


Figure 3: Colorbox showing a set of 5 pictures

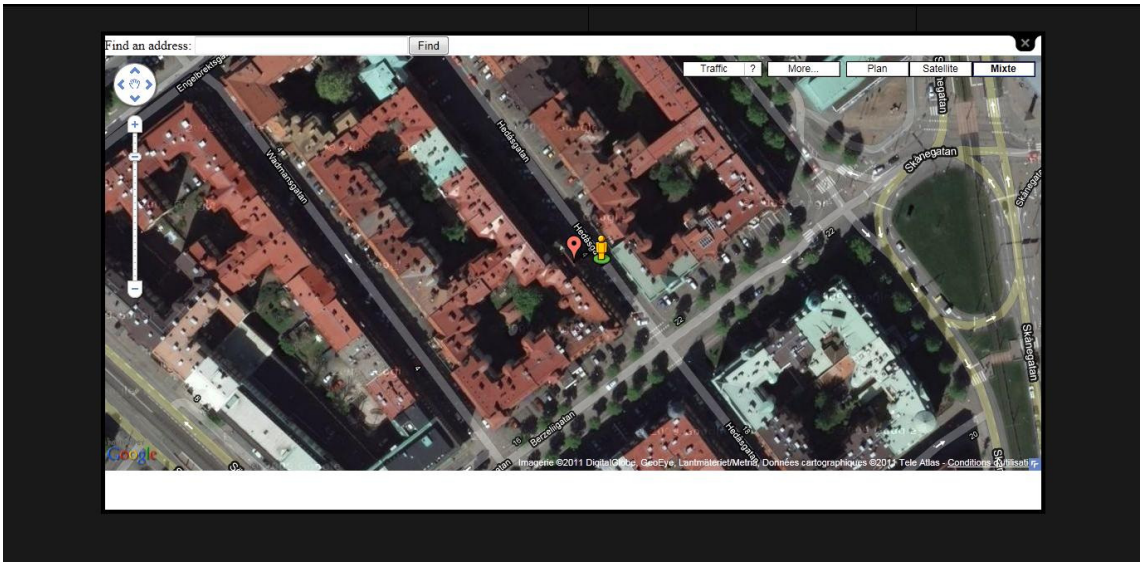


Figure 4: Colorbox showing the property's location on a map

Problem 2:

After implementing the above solution, we noticed that it wasn't as efficient as we would have wanted. Indeed, a user doesn't need only to read information about the property, but handle it as well. He may want to copy some of the text given, and the tooltip feature doesn't allow that. Also he may want to send an email to the property' owner, in which case he'd be forced to go to the property's page to do so. This will render our approach and effort to ease the user's search unfulfilling and incomplete.

The naïve solution that comes to mind would be to use popup boxes like it is done for displaying images and the property's map. However this could prove a little "messy" and unprofessional.

Investigation:

We implemented a few solutions that ended in unwanted results. Thus the Investigation lead to ruling out the following options:

- Using a tooltip: since it is not possible to copy information from it
- Using a tooltip that closes manually: it may prove to slow down users even more.
- Using a popup box (as it is done for images): it would be simpler to just visit the property's ersonal page. It would also give off an unwanted feel that the website is not homogenous.
- Using Jquery based solution since it would consume considerable CPU for even a standard number of result listings.

The solution needs to be simple, and integrates well with the overall look and feel of the website. We want to reduce clicking to access the information to the maximum. Thus, the outcome of the investigation was to follow the same model for displaying pictures, to display property's information. We replace the sliding pictures with sliding panels as follows:

Figure 5: initial view of the property's information from the results page

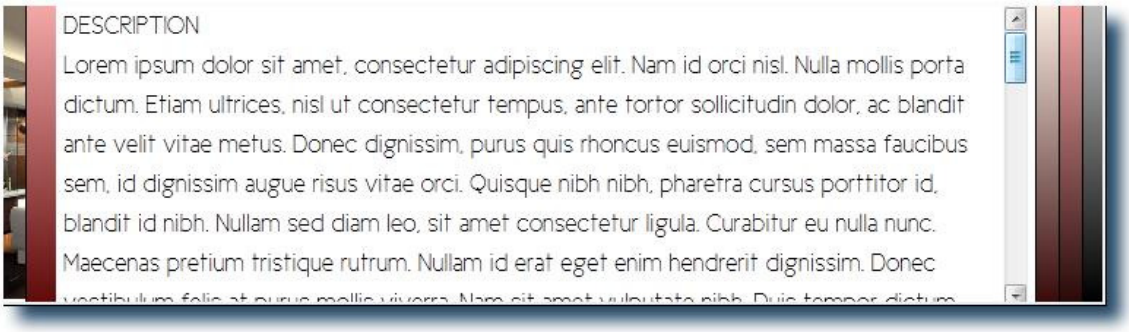


Figure 6: showing the slide containing the property's description

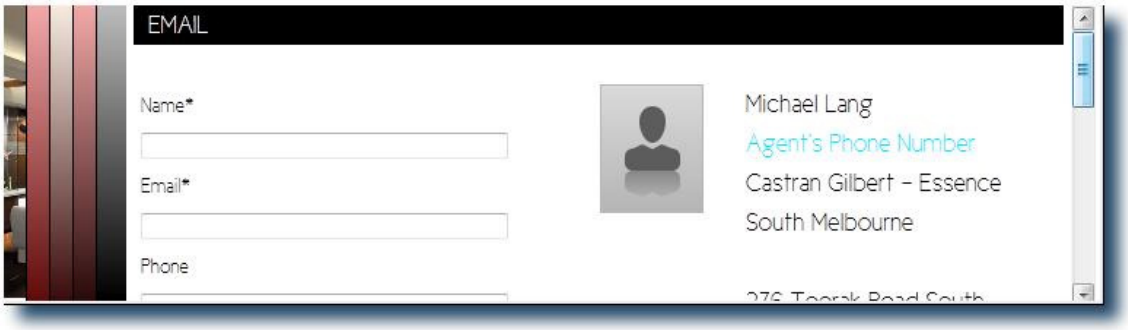


Figure 7: Using the last slide to send email

We can notice that when a slide content is too long, it's possible to scroll up and down to view all content.

5.4 Email from results page

Problem

When sending an email to a property owner, the user tends to always fill in the same message content, namely:

- The client's name, last name, email, and personal information
- The client's comments

Therefore, it is mandatory to save this information for the user and reproduce it for every email form. In this manner, the client can send considerably many more emails than otherwise if needed.

Investigation & Solution

Basically, the investigation led to the following two potential solutions:

- To Use Javascript to copy any content the user input inside one email form to all other forms.
- To fill in the content of the email form at the server side at page load

The optimal solution was chosen to be the second one since the page is reloaded at every email submission. We can then use this to save the user data statically at the server side, and then fill in the form fields with it before page load. Naturally, the data will change everytime the user sends an email, and the new data will be used to fill in the blanks.

The code would look as follows:

```
//Result.aspx /*html page*/

<label for="fromName">
    <strong>Name<span>*</span></strong>
</label>
<div>
    <input runat="server" id="fromName" name="fromName" type="text"  value=""/>
</div>
…
<input runat="server" id="likeTo1"type="checkbox" value="Get an indication of price"/>
<label for="MainContent_likeTo1">Get an indication of price</label>
…
<label for="message">Comments</label><br />
<textarea runat=”server” id="message" name="message" >
</textarea>
…
```

```
/Result.aspx.cs /*code page*/

public partial class Result : System.Web.UI.Page
{
      static string fname = "";
      //...
      static bool liketo1 = false;
      //...
      static string msgtext = "";
      protected void Page_Load(object sender, EventArgs e)
      {
          fromName.Value = fname;
          //...
          likeTo1.Checked = true;
          //...
          message.InnerHtml = msgtext;
      }
}
```

5.5 Property presentation

Even though a user can get all information he may require just from the results page, it was mandatory to keep the tradition of having a specific dedicated page for every property.

As for the layout we had two options:

- Use a simple page as it is done in all website, where on top, picture of the property can be found, while at the very bottom, there's an email form.
- Use the sliding panels as it is done in the results page, only this time with larger space (width and height).

We opted to use both the two solutions. A registred user can chose his default display. However, due to time contraints, we decided for the time being, only the first option will be implemented. It is our way not to stray the user too far from what he is used to which would make him feel uncomfortable and drive him away from our website.

5.6 Quick Interaction with results

Problem 1

In most cases, the user needs to interact directly with a property listing from the results page. For example, if the user wants to add a property to his selection, it would be a hindrance to have to go to the property's page first. Instead he should be able to do it directly from the results page in the same way we provide him with the ability to view all the property's information from within the results page.

The investigation then comprises of finding what links may be usefull for the user, and ways to implement them. Such user interactions are as follows:

- Share property with a friend on Facebook
- Post property to your facebook page
- Send property to a friend's email
- Add property to user's selection
- Print the offer
- Send simillar offers by email

Solution

These links will be displayed next to the property's pictures on the first slide at the results page. As for the functionality implementation, consider the following example:

We want to allow the user to add a property to his property selction. For this we chose to create a specific page to handle this task as follows:

- The "add to my selection" link will point to "addproperty.aspx" page:

```
<a href="addproperty.aspx?add=PROPERTYID&previouspage=PREVIOUS-PAGE-LINK”>Add to my selection</a>
```

- The PROPERTYYID is, as the name describes, the id of the property, whereas the PREVIOUS-PAGE-LINK is the link to the results page that the link was clicked at.
- At the "addproperty.aspx", we add the id to the (logged-in) user's selection table in our database, and redirect back to the previous results page.
- At the results page, there should no longer be an "add to my selection" link. For this reason, when loading the results page, we check to see if there are any displayed results already figuring inside the user's selection:

```
protected void Page_Load(object sender, EventArgs e)
{
  …
  ArrayList selectedids = new ArrayList();
    // query “UserSelection” table for all entries for this user
    …
    // display property with identity = id
    // check if this property is already selected by the user
  if (selectedids.Contains(id))
  {
     listings.InnerHtml += "<a href='selection.aspx'>Already in My Selection</a><br/>";
  }
  else
  {
      listings.InnerHtml += “<a href= ‘addproperty.aspx? add=”+ PROPERTYID
            +”&previouspage=”+ PREVIOUS-PAGE-LINK +”'>Add to my selection</a>”;
  }
  …
}
```

Remark

Only a few of the aforementiened useful links were implemented while the rest was decided to have less priority and to be implemented at a later time.

5.7 Two Results listing display layouts

Problem

In the layout we are proposing, the user can only view up to 3 property listings in his screen at once. Therefore, it was mandatory to provide an alternative layout where more results could be viewed.

Investigation

Once again, we tried to define the best way an alternative display for our search results listings, without straying too far from the traditional way it is done in most wesbites. Our options were:

- Present the results in a table form (as it is done in boplats.se)
- Use a grid view
- Use picture carousel

Unfortunately, even if the last option could be attractive to the user, it would require a lot of time to implement, and it might not be the ideal display choice for a real estate website. Instead we chose to use the grid view as our alternate display. It is the same display used recently by blocket.se. We also decided to have the two default tile sizes that the user can choose from. So at the end, he’s capable of seeing a minimum of 6 property’s at a time, and up to 15 for the second default size.

Overall, the results display look as follows:

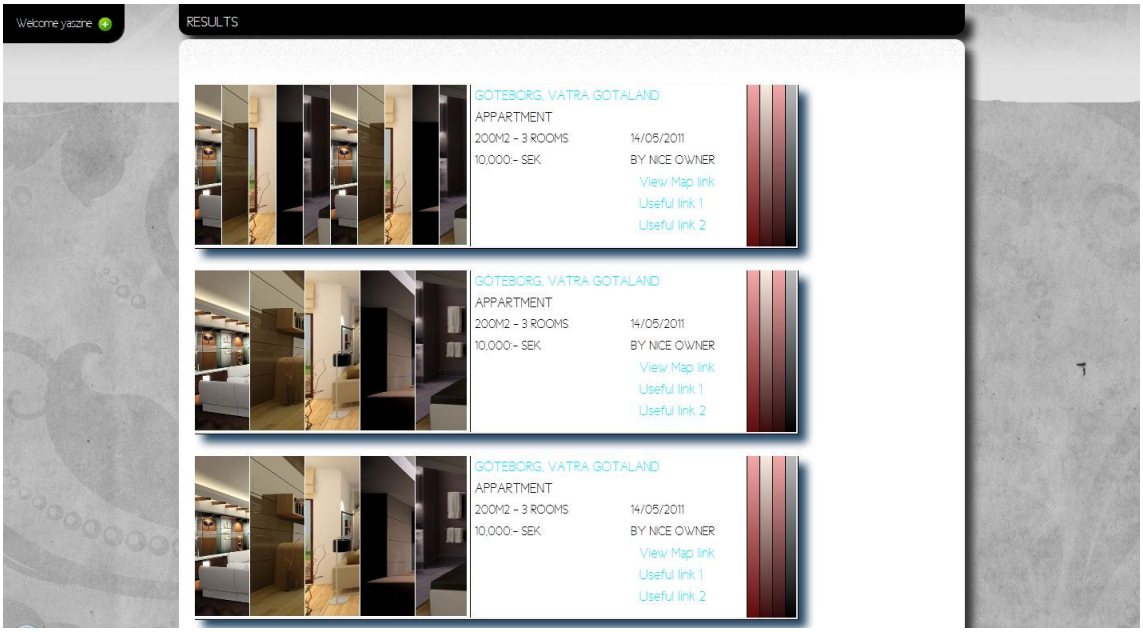


Figure 8: Default Results display: the detailed option

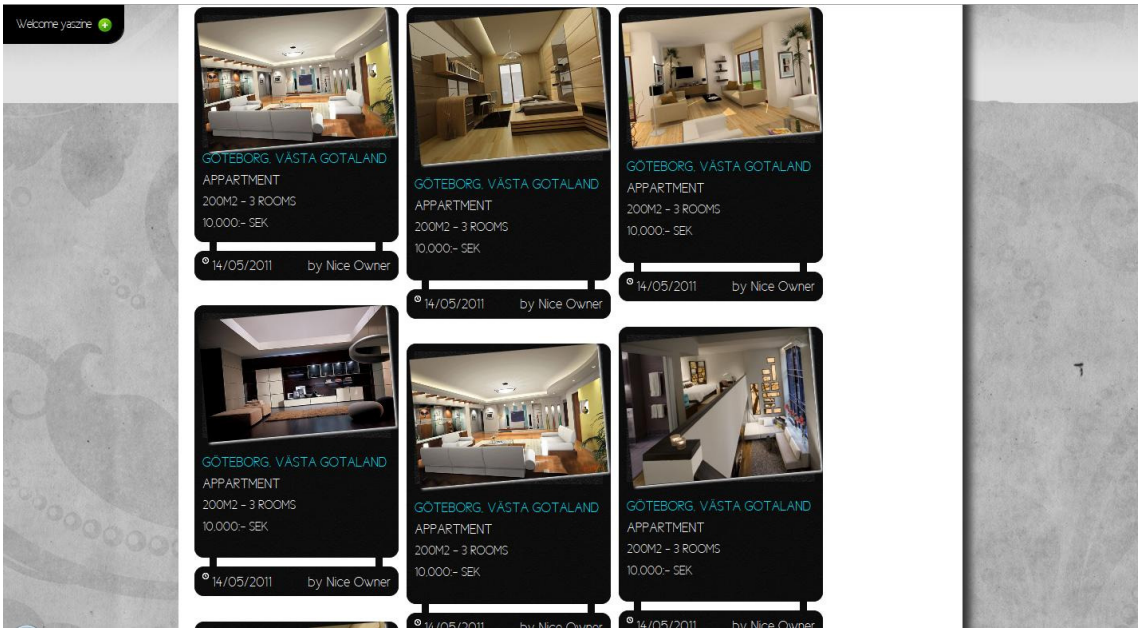


Figure 9: Grid view in a large size tile format.

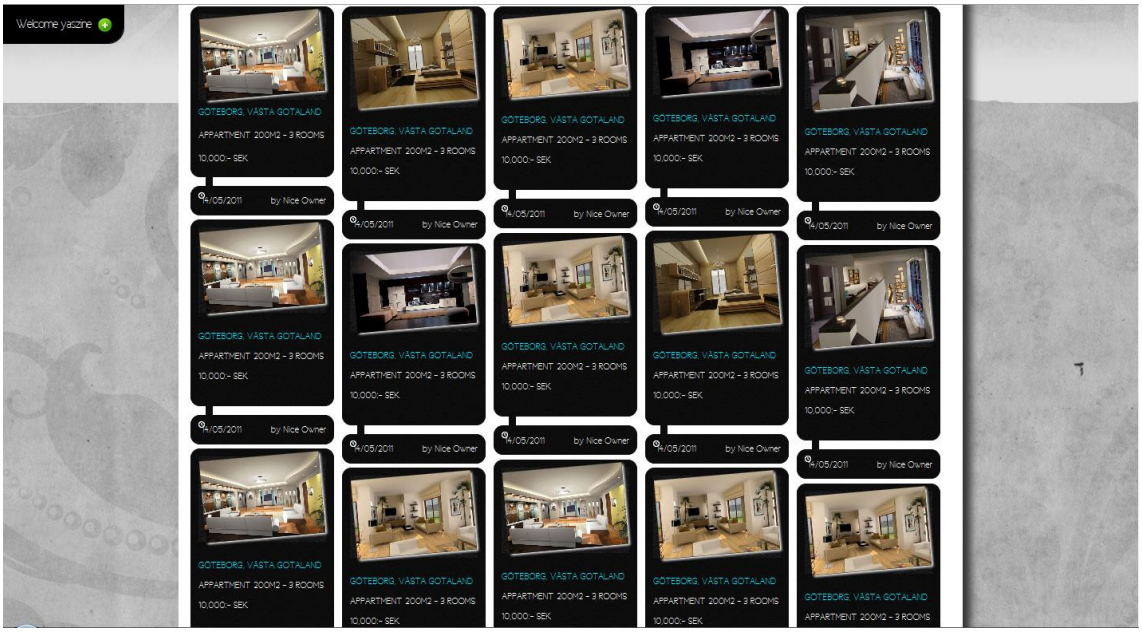


Figure 10: Grive view In a smalle size tile format

In The small tile format, we thought of tilting the pictures for a friendlier look and feel. Also throughout the design, we aimed to use modern techniques and colors to appeal to the customer, and give the website a hint of professionalism. This was achieved through the use of light and dark shadows, round corners, an innovative design, and an overlaping “brick” layout for our tiles.

The overlapping brick layout was done using jquery, and more precisely the “`jquery.masonry.min.js`” library: [12]

- first we create a “container” tag
- within this container, we write down our “brick” code.
- Our brick code is made of many “brick” tags that each display one property’s information as shown in figure 10 and 9.
- Then at the end, we initialize the jquery function responsible for the given brick display.

Example:

```
<div id="container">
            <div class="brick">
            …
              </div><!-- end .brick -->
              <div class="brick">
            …
              </div><!-- end .brick -->
            …
</div><!-- end #container -->
            …
```

```
<script src="js/jquery.masonry.min.js"></script>
<script>
    $(function () {
        var $container = $('#container')

        $container.imagesLoaded(function () {
            $container.masonry({
                itemSelector: '.posts'
            });
        });
    });
</script>

            …
```

Each brick content and styling is up to the web engineer to define, and it could be anything such as simple images or plain text.

Problem 2:

Introducing the extra types of display layouts to our website's results listings does indeed enable the user to view more properties. However, they also seem to restrict his ability to view more information about the property directly from the results page. Indeed, the aim of our website is to hasten the user's search. And for this reason, we aim to find even more possible ways to make our result display rich of information. The difference between the amount of information displayed and ease of access we provide in the first standard layout, is very apparent. Whereas, even though the second displays are made to solve a problem that may reside in the first layout, it holds very little useful information or options. Nonetheless, our through investigation and brain storming ended up in choosing the use of a tooltip as the only feasable solution.

Investigation

We are very limited by the space each ad provides. As a matter of fact, the small space each property is allocated in our results display renders any possible solution either hard to implement or useless:

- Flipping the property display's block. At the backside, we could put the property's description and features. However the space is too small for that which would make this option of no purpose nor use.
- Displaying the property's personal page within the current one as it is done for the pictures and map. However it may be simpler to just let the user visit the property's page.
- Embed the first layout within the second. Hovering over a property will trigger the sliding panels to be shown at it's side. This in turn can prove to be once again useless, since it's simpler to visit the property's page.
- Dedicate a special area at the side of the display whose content changes dynamically when hovering over a property. The area's content will contain all property's

information. However, this solution was dicarded since the area will limit the layout purpose to display as many properties as possible within the same window screen.

Therefore, the only plausible solution would be to use a tooltip. To avoid any user discomfort that may result from the tooltip, we decided to add a little vertical bar to the side of the property's brick. Hovering over the vertical bar would trigger the tooltip. In this manner, the tooltip will most likely show only when the user wants to display it, rather than showing by mistake and present a nuisance to the user's browsing experience. [8]

5.8 User pages and Side panel

In most cases, in a real estate website, signed up users should have their own space. However, how to build or present such pages is not trivial, and there is much room for improving the way user pages are implemented. After brief discussion, we decided it would be best to have a side sliding panel to the side, where the user can login, and access his pages quickly. It would consitute his own personal side menu.

The traditional way of implementing the user pages is through providing the User with his own personal page that links to other pages. We judged this method to be perfectly satisfactory and gathers to all user needs. Nonetheless, in our quest to do things better, we thought that the same area used for user login, should also be used to display links to his pages. It would give off the feeling of being a social website, a little like Facebook. It would make the user feel that the website is his, that he's welcome to use as often as he wants, and that it is made for him, and him only.

In other websites, the user would still feel like it's just any other search space. But in our setting, we try to make it a little more personal.

5.8.1 Annonymous View

When the user is not authenticated, the sliding panel will display a login field requesting the user to login. Putting the login inside a sliding panel allows for a quick login anytime the user wants to. We also understand that a user's search for a property to rent or buy can not be done in one day. And for this reason, we provide the feature of remebering the user's login credentials for future logins so he'd be signed-in anytime he visits the website.

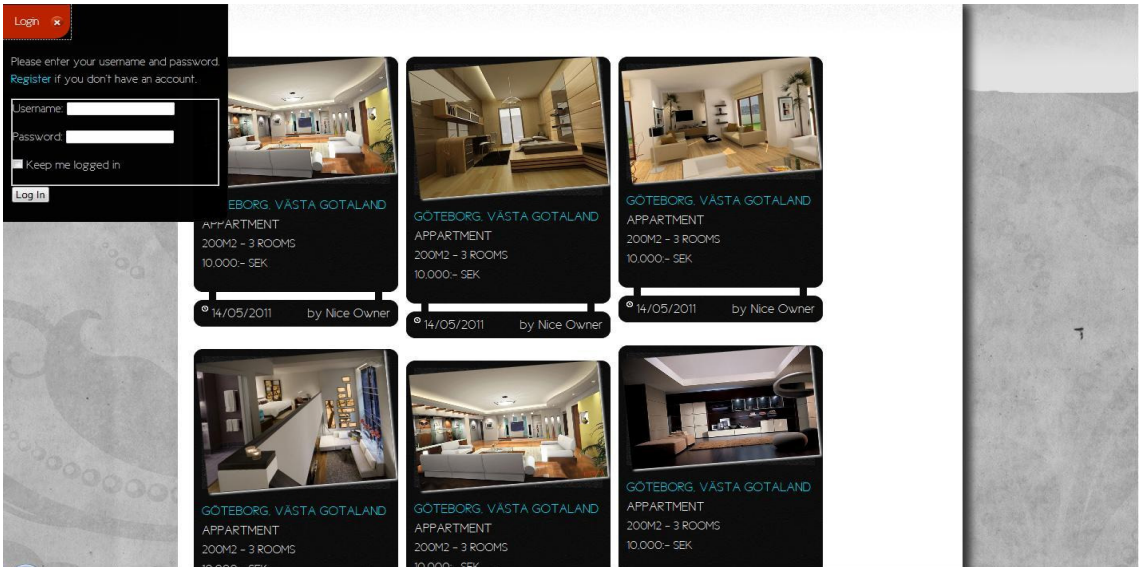


Figure 11: expanded side panel before User authentication.

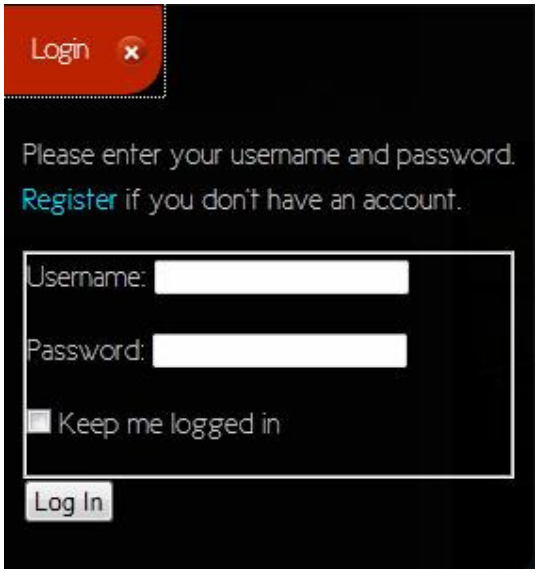


Figure 12: side panel upclose showing authentication field.

5.8.2 Logged-in view

When the user is authenticated, the panel content changes to a menu containing his personal pages as the following figure shows:

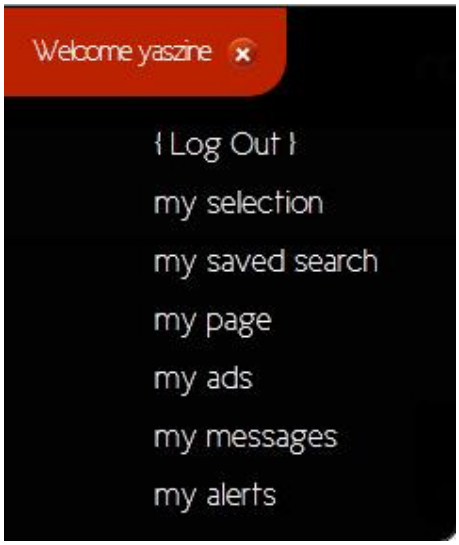


Figure 13: side panel with the user personal pages

The side panel allows the user a quick access to his saved search, so he could quickly pick up from where he left from before.

Figure 14: The user has quick access to his saved search directly from the side panel.

To change between the two views of the side panel (the anonymous view, and the authenticated view), we utilize ASP's LoginView. This is done as follows:

- At the site.master.aspx page, we add the following code

```
<asp:LoginView ID="LoginView1" runat="server" EnableViewState="false">
    <AnonymousTemplate>
        <div class="panel">
            <span id="insidesearch" runat="server"></span>
            <h2>
                Log In
            </h2>
            <p>
                Please enter your username and password.<br />
                <asp:HyperLink NavigateUrl="Account/Register.aspx"
                    ID="RegisterHyperLink" runat="server"
                    EnableViewState="false">
                    Register
                </asp:HyperLink>
                if you don't have an account.
            </p>
            <asp:Login ID="LoginUser" runat="server" EnableViewState="false"
                RenderOuterTable="false">
                <LayoutTemplate>
                    <span class="failureNotification">
                        <asp:Literal ID="FailureText" runat="server"></asp:Literal>
                    </span>
                    <asp:ValidationSummary ID="LoginUserValidationSummary" runat="server"
                        CssClass="failureNotification"
                        ValidationGroup="LoginUserValidationGroup" />
```

```
<div class="accountInfo">
   <fieldset class="login">
         <p>
            <asp:Label ID="UserNameLabel" runat="server" AssociatedControlID="UserName">
            Username:
            </asp:Label>
            <asp:TextBox ID="UserName" runat="server" CssClass="textEntry">
            </asp:TextBox>
            <asp:RequiredFieldValidator ID="UserNameRequired" runat="server"
            ControlToValidate="UserName" CssClass="failureNotification"
            ErrorMessage="User Name is required." ToolTip="User Name is required."
            ValidationGroup="LoginUserValidationGroup">*
            </asp:RequiredFieldValidator>
          </p>
          <p>
            <asp:Label ID="PasswordLabel" runat="server" AssociatedControlID="Password">
            Password:
            </asp:Label>
            <asp:TextBox ID="Password" runat="server" CssClass="passwordEntry"
            TextMode="Password">
            </asp:TextBox>
            <asp:RequiredFieldValidator ID="PasswordRequired" runat="server"
            ControlToValidate="Password" CssClass="failureNotification"
            ErrorMessage="Password is required." ToolTip="Password is required."
            ValidationGroup="LoginUserValidationGroup">*</asp:RequiredFieldValidator>
          </p>
          <p>
            <asp:CheckBox ID="RememberMe" runat="server" />
            <asp:Label ID="RememberMeLabel" runat="server"
            AssociatedControlID="RememberMe" CssClass="inline">
            Remember me
            </asp:Label>
           </p>
   </fieldset>
           <p class="submitButton">
            <asp:Button ID="LoginButton" runat="server" CommandName="Login"
            Text="Log In" ValidationGroup="LoginUserValidationGroup" />
           </p>
</div>
```

```
     </LayoutTemplate>
   </asp:Login>
 </div>
 <a class="trigger" href="#">Login</a>
</AnonymousTemplate>
```

```
<LoggedInTemplate>
        <div class="panel" id="mypanel">
            <ul id="nav">
                <li><a href="~/"><span>
                    <asp:LoginStatus ID="HeadLoginStatus" runat="server"
                         LogoutAction="Redirect" LogoutText="[ Log Out  ]"
                         LogoutPageUrl="~/" />
                </span></a></li>
                <li><a href="Selection.aspx">my selection</span></a></li>
                <li><a href=""><span>my saved search</span></a>
                    <ul id="listitems" runat="server"></ul>
                </li>
                <li><a href="profile.aspx"><span>my page</span></a></li>
                <li><a href=""><span>my ads</span></a></li>
                <li><a href=""><span>my messages</span></a></li>
                <li><a href=""><span>my alerts</span></a></li>
            </ul>
        </div>
        <a class="trigger" href="#">Welcome <span class="bold">
            <asp:LoginName ID="HeadLoginName" runat="server" />
        </span></a>
    </LoggedInTemplate>
</asp:LoginView>
```

- Then we use the jquery library to animate our sliding panel with the following code:

```
<script type="text/javascript" src="js/jquery.min.js"></script>

<script type="text/javascript">
    $(document).ready(function () {
        $(".trigger").click(function () {
            $(".panel").toggle("fast");
            $(this).toggleClass("active");
            return false;
        });
    });
</script>
```

- The styling for the panel can be done as follows:

```
.panel {
   position: fixed;
   top: 0px;
   left: 0;
   display: none;
   background: #000000;
   border:1px solid #111111;
   -moz-border-radius-topright: 0px;
   -webkit-border-top-right-radius: 0px;
   -moz-border-radius-bottomright: 10px;
   -webkit-border-bottom-right-radius: 10px;
   width: auto;
   height: auto;
   padding: 40px 10px 10px 10px;
   filter: alpha(opacity=95);
   opacity: .99;
   z-index: 990000;
}
```

```
.panel p{
    margin: 0 0 15px 0;
    padding: 0;
    color: #cccccc;
}

a.trigger{
    position: fixed;
    text-decoration: none;
    top: 0px; left: 0;
    font-size: 16px;
    letter-spacing:-1px;
    color:#fff;
    padding: 10px 40px 10px 15px;
    font-weight: 400;
    background:#000000 url(images/plus.png) 90% 55% no-repeat;
    border:1px solid #000000;
    -moz-border-radius-topright: 0px;
    -webkit-border-top-right-radius: 0px;
    -moz-border-radius-bottomright: 20px;
    -webkit-border-bottom-right-radius: 20px;
    -moz-border-radius-bottomleft: 0px;
    -webkit-border-bottom-left-radius: 0px;
    display: block;
    z-index: 990000;
}

a.trigger:hover{
    position: fixed;
    text-decoration: none;
    top: 0px; left: 0;
    font-size: 16px;
    letter-spacing:-1px;
    color:#fff;
    padding: 10px 40px 10px 15px;
    font-weight: 700;
    background:#000000 url(images/plus.png) 90% 55% no-repeat;
    border:1px solid #000000;
    -moz-border-radius-topright: 0px;
    -webkit-border-top-right-radius: 0px;
    -moz-border-radius-bottomright: 20px;
    -webkit-border-bottom-right-radius: 20px;
    -moz-border-radius-bottomleft: 0px;
    -webkit-border-bottom-left-radius: 0px;
    display: block;
    z-index: 990000;
}

a.active.trigger {
    background:#b20 url(images/minus.png) 90% 55% no-repeat;
    z-index: 990000;
}
```

- Finally to load the user specific saved search, we use the following page-load code at the site.master.cs file (if the user is authenticated):

```
using (SqlConnection conn = new SqlConnection(
             ConfigurationManager.ConnectionStrings["ApplicationServices"].ToString()))
   {
                SqlCommand cmd = new SqlCommand();
                cmd.Connection = conn;
                cmd.CommandText = "SELECT * FROM userSearch WHERE Username = @Username ";
                cmd.CommandType = CommandType.Text;
                cmd.Parameters.Add("@username", SqlDbType.NVarChar).Value =
                        this.Page.User.Identity.Name;
                conn.Open();

                HtmlGenericControl mylist =
                        (HtmlGenericControl)this.LoginView1.FindControl("listitems");

                using (SqlDataReader reader = cmd.ExecuteReader())
                {
                    if (reader.HasRows)
                    {
                        while (reader.Read())
                        {
                            if (mylist != null)
                            {
                                mylist.InnerHtml += "<li><a href='" +
                                      Convert.ToString(reader["Search"]) + "'>" +
                                      Convert.ToString(reader["Date"]) + "</a>" +
                                      "<ul><li><a href='" +
                                      HttpContext.Current.Request.Url + "&remsearch=" +
                                      Convert.ToString(reader["id"]) + "'> (remove) </a>
                                      </li></li></ul></li>";
                            }
                        }
                    }
                    else
                    {
                        if (mylist != null)
                        {
                            mylist.InnerHtml = "<li><a href='Search.aspx'>Start A new
                                      Search</a></li>";
                        }
                    }
                }
   }
```

5.9 Database technicalities

To return a search results, we are faced between two options:

- Query the entire database and return ALL matching entries, which are then paged at the client side.
- Or perform paging when quering the database at the server side.

For simplicity, we chose the second option to avoid any plausible delays that may be introduced from large database querying. We use a stored procedure that is reused throughout our code to ease any further database queries.

The procedure takes 6 arguments:

- The table to query
- The order by which we want the result displayed

- The Where statement to filter our results
- The page number
- And the page length

Example:

```
SqlCommand cmd = new SqlCommand("utilPAGE",conn);
cmd.CommandType = CommandType.StoredProcedure;
…
cmd.Parameters.Add(new SqlParameter("@datasrc", "Ads"));
cmd.Parameters.Add(new SqlParameter("@orderBy", ordercmd ));
cmd.Parameters.Add(new SqlParameter("@fieldlist", '*'));
cmd.Parameters.Add(new SqlParameter("@filter", cmdtext));
cmd.Parameters.Add(new SqlParameter("@pageNum", page));
cmd.Parameters.Add(new SqlParameter("@pageSize", pagelength));
```

5.10 Comprehensive upload form

A comprehensive upload form can play a big role in enhancing the property's advertisement process. In most cases, letting the property owner or the real estate agent handle all of the property's description on his own proves to be unfulfilling, and very limiting. It is then our responsibility to help the property's owner to successfully advertise his property in its full image.

Investigation:

In this projuct phase, our main aim is to understand what a real estate needs to fully describe the property he's advertising, and provide him with the means to do so. We tried to find the most relevant features that may be needed to describe a property and divided them into two categories: Interior Features and Exterior features. Their respective content is as shown in the following figure:

Figure 15: Upload form features section

We also need to provide the user with the possibility to upload as many pictures as he can. We achieve this by allowing the client to add as many upload fields to upload his pictures in dynamically. We use javascript for this purpose as follows:

```
<label>Pictures: </label>
<p id="upload-area">
    <input class="s1" id="File1" type="file" runat="server" size="60" />
</p>
<input class="s1"  id="AddFile" type="button" value="Add file"
            onclick="addFileUploadBox()" />
```

```
<script type="text/javascript">
                function addFileUploadBox() {
                    if (!document.getElementById || !document.createElement)
                        return false;
                    var uploadArea = document.getElementById("upload-area");

                    if (!uploadArea)
                        return;

                    var newLine = document.createElement("br");
                    uploadArea.appendChild(newLine);

                    var newUploadBox = document.createElement("input");

                    // Set up the new input for file uploads
                    newUploadBox.type = "file";
                    newUploadBox.size = "60";

                    // The new box needs a name and an ID
                    if (!addFileUploadBox.lastAssignedId)
                        addFileUploadBox.lastAssignedId = 100;

                    newUploadBox.setAttribute("id", "dynamic" +
                    addFileUploadBox.lastAssignedId);

                    newUploadBox.setAttribute("name", "dynamic:" +
                    addFileUploadBox.lastAssignedId);

                    uploadArea.appendChild(newUploadBox);
                    addFileUploadBox.lastAssignedId++;
                }
</script>
```

5.11 Country and language selection

Since the website we're implementing is meant to be international, it is necessary to provide the possibility to select a specific country, and an appropriate language for that country.

For this, we try to detect the user's ip address and directly forward him to his respective country's website folder. The user can then, if he wishes, select another country. When he clicks the "change country" link, he's forwarded to the country selection page. However, since right now, we're focusing mainly on Sweden, and no other websites, this feature is on hold, and will be used when the website is available in more countries.

Nonetheless, we should be mindful that even if the user is using his host country's website, he may not speak the language in which data is shown by default. For example, the Swedish website will have two default languages: English and Swedish. Therefore, to view the website content in an other language of the user's preference, we use the google translation gadget. This is done by embedding the following google code into our website [10]:

```
<div id="google_translate_element"></div>
<script>
                function googleTranslateElementInit()
                {
                                new google.translate.TranslateElement({
                                pageLanguage: 'fr'
                }, 'google_translate_element');
}
</script>
<script src="//translate.google.com/translate_a/element.js?cb=googleTranslateElementInit">
</script>
```

5.12 Featured properties display

In this section, we researched different ways to display featured properties in our homepage. Indeed, we found out it could be done in numerous ways. But our main focus was to use an image slider across the the whole web page. For that, we had the following options:

- Using a flash application, or more precisely the “cuber Image slider”.
- Build our own flash application, but it will cost us a great amount of time.
- Use Jquery to implement an intuitive layout, or search existing ones.
- User CSS3 slider as it is done for our image display. However, this doesn’t fit our purpose which is to have the images change dynamically.

Both the flash and jquery based sliders, have many varieties and they all seem to fullfill our criterias. So we simply picked the most appealing to us as shown below.

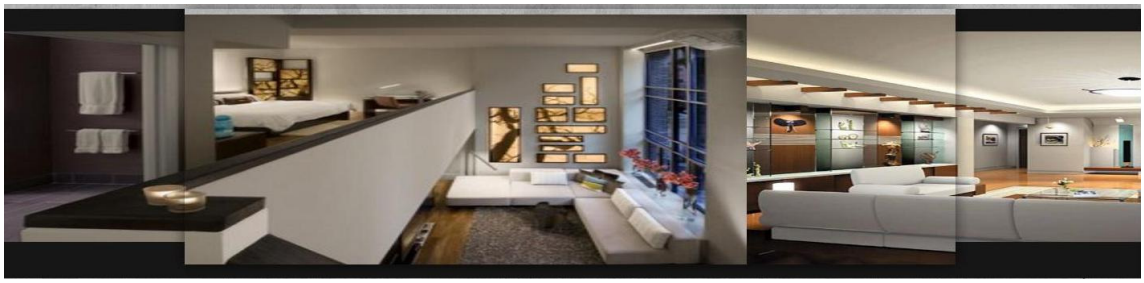


Figure 15: horizontal slider across the website page.

5.13 Search by map

In our website, we also include the search by map feature. The properties are marked on the map via a marker. Clicking the marker displays the basic property’s information (price, property type, and surface area) with a link to the property’s personal page. However, the search is done at the server side, rather than the client side. Thus, the results seem static rather than continuously changing as the user moves across the map. We thought this would be better since it allows the user to see all properties corresponding to his search criteria at

once, and it doesn't require time to reload everytime the user moves to a different map area.

The search form for this feature is the same as our basic search. The only difference is that results are displayed all within the map rather than in a list form.

A future improvement to the feature would be to add an extra result display layout in which the user can see both the property information, and it's location on the map simultaneously.

5.14 Help links and News

When a client is looking for a place to stay or move to, he's usually looking for more than just that. We want to provide the user with all what he needs to successfully find a place and move in to it. For this reason, we thought of providing help links and information related to moving. We investigated possible help we could provide for prospective users. In the following a few of the help capabilities we provide are listed:

- Help links for students about registration and such.
- Contract types or models.
- Links to mover companies.
- Links to furniture design, and local furniture stores.
- Property purchase tips.
- Loan information, tips, and calculator.
- Local Attraction places.
- Security links and law information
- Community provided space where users can share links and information.

6. Examples

In the following, a few screenshots of the website created so far are presented. We should note that most of the functionalities (map facility, user utilities, upload form…) and section of pages were discussed during the navigation section. Therefore, only a general overview of how the website looks like is shown here.

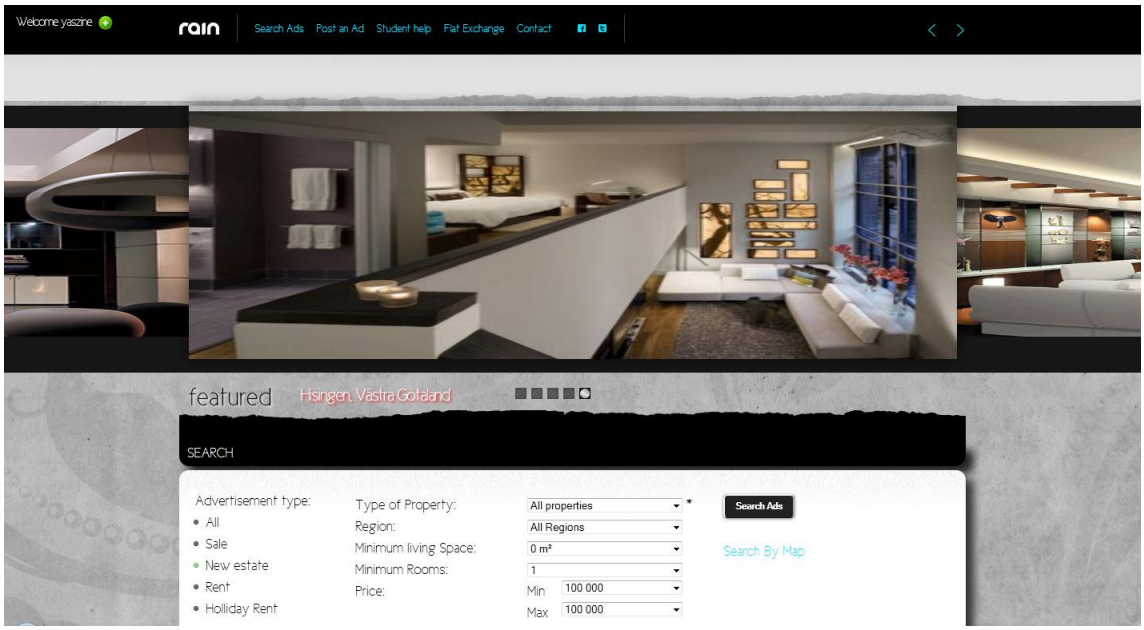


Figure 16: Website Home page with search form. In the top of the page, we find the featured properties displayed wide across the web page, and dynamically changing every 4 seconds.

Figure 17: Possibility to sort results by date, price, or surface area, in both ascending and descending order. It is also possible to save the search results into the user's saved searches

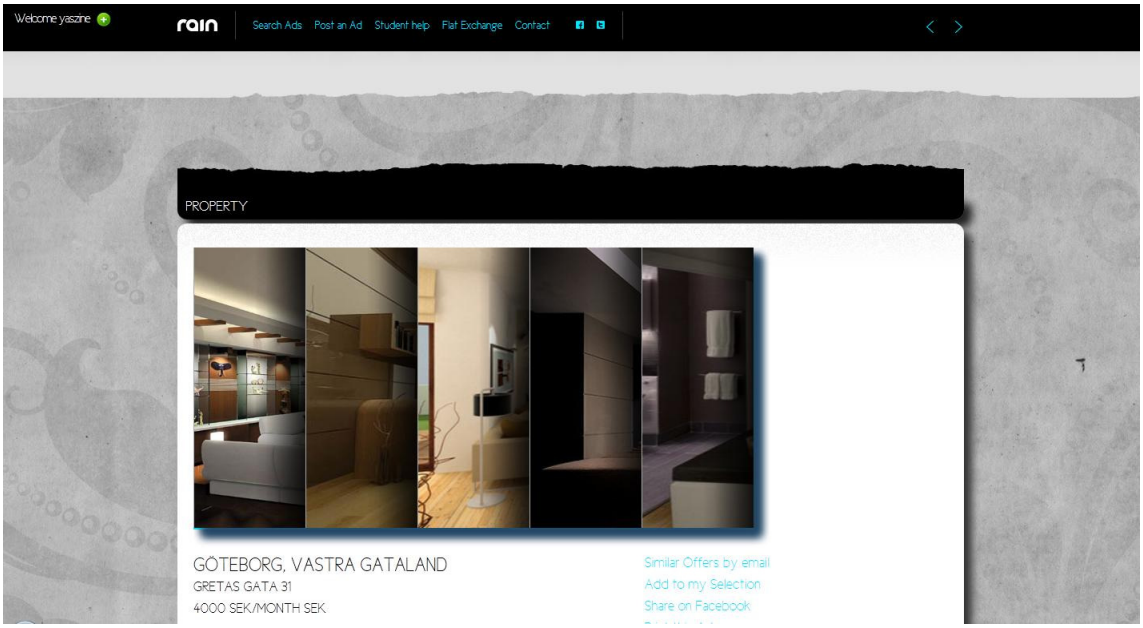


Figure 18: The personal page for each property. At the top can be seen the sliding pictures of the property, when right beneath the pictures, information about the property along with useful links are displayed. Clicking on a picture will trigger it to be displayed in full size. The space to the right was left blank in purpose to host paid ads.

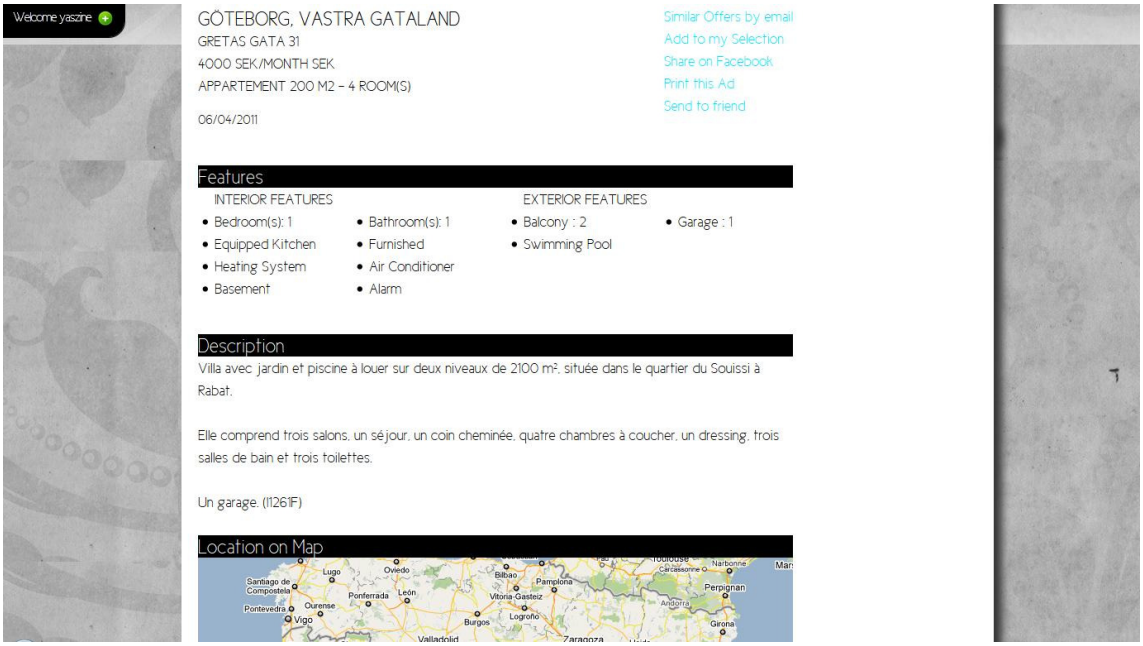


Figure 19: continuation of the property view where we can read about the features of the property, its description, and locate it on a real map.

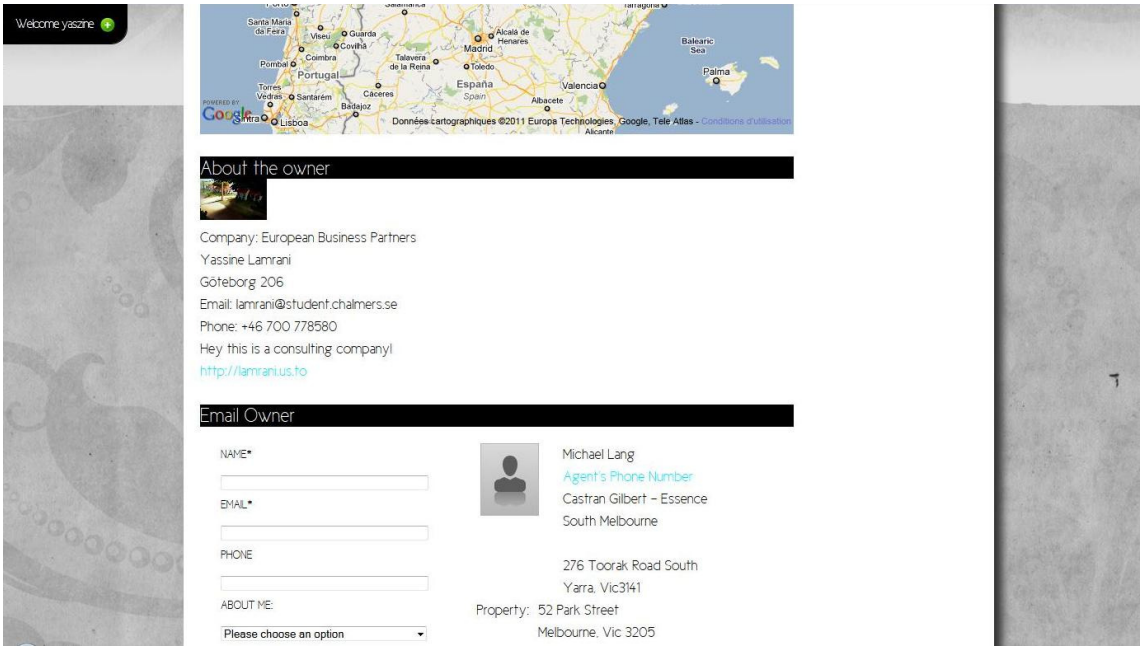


Figure 20: continuation of the property view page. At the bottom of the page, we find information about the owner (its name, logo, and website if applicable,…), along with an intuitive email form.

7. Comments

The design chosen for the website is very much subject to change. After all, the website is still in the trial period. Indeed, even its functionalities are meant to continue growing throughout time. As a matter of fact, my sponsors and I feel the main weakness of the other websites is that they lack continuous update and improvement. In this same manner, we also feel our own website needs to grow even further to reach an internationally professional level and appeal to both realtors and normal users. And in order to do so, we need to criticize our own work constantly to never fall behind. We can affirmatively claim that the website is not yet at the international level, yet, we would also like to believe it is well on the track to be so. We started from scratch, with a mere idea, a motivation and simply a lot of investigation to undergo. We started with no specific design in mind, and simply the need to make it look modern. So what is achieved so far is yet just the beginning of the embodying of our ideas and motivation. And as I continue my work in implementing further requirements they may have, the shape of the website is sure to differ from how it is now.

The website as it is now, still needs a lot of content. A real estate website is not just about posting ads, and searching through them. Indeed, any website we try to relate to can be found to have many side links. Whereas, the main strength of such a website is both the features and capabilities it offers. Design seems to always not matter as much. Maybe the user wishes for a better looking website, but so far the mostly used ones fall short on that aspect. This leads us to the conclusion that what's more important than the design is the content, and most of all the number of users using the website. The hardest part is not the coding, but rather attracting customers. Marketing then plays the biggest role in promoting such website. Nonetheless, without something great to promote, it is doomed to fail.

What we offer now certainly still has many weaknesses but as well many strengths of its own. It can stand its ground inside a competitive market like today's. Also, according to marketing professionals, if we consider the fact that it takes sometimes up to 4 years to promote a normal website and create a brand for it, a real estate website takes much longer. What has been implemented so far already comprises of innovative presentation techniques and core features. On one hand we still need more to make it close to perfect, but on the other hand, we don't expect to have it all done as a 3 months thesis project. In fact, the sponsor company expects the project to grow further beyond the thesis time frame. Indeed, the main purpose of the thesis could be said to lay down the necessary ground work (Investigations, and beta implementation) for the creation of the website.

The following chart describes the weaknesses and strengths of the website

Strengths	Weaknesses
▪ Modern design ▪ Quick and easy usage ▪ Simple and fresh	▪ Less content ▪ A few capabilities offered in other websites are yet to be offered in ours

▪ Still growing ▪ New technical aspects	

Most of the website's weaknesses will be addresses in our future work. But what we focused on mostly is to attract new customers for now. After all, what use would it be to spend a lot of time on implementing a highly sophisticated search form when there's only but a few properties to search from anyway? Such things take second priority. It is in every customer to like "shiny pretty" things. And so the company's objective was and still is: to make the website clean, pretty and shiny. In fact, in the short time we had, we needed to do and redo most of the website's component for this very purpose. Because at the end, you can't know how well something is until you try it. Similarly, the company also aims to try the current version of the website first on the market even if it may be a little far from finished. We want to have the user's feedback and know what he wants first before we can give it to him. In this way, we wouldn't invest and waste time and hard work giving the user something of little interest to him.

8. Future work

8.1 Immediate development

For the time being, the first additions to be made on the website would be as follows:

- Upload it on a server.
- Populate it with room ads from the Goteborg area so the upcoming students for the fall 2011 could already start using it.
- Provide all possible help links for students
- Translate the website to Swedish.

Then the website will be ready for shipment. The first advertisement target to which the website will be aimed at will be universities in the Gothenburg area. Afterwards, the website will further undertake the following immediate additions:

- Add more help links
- Populate the website with more adds from the whole of Sweden
- Add different purpose calculators (for loan and such)

At this point, the website will be advertised at the company's monthly magazine for other entrepreneurs to look at.

Technically, all further development of the website would be to provide more search options and ease for the customer, and more appeal to the real estate agent. In fact, it is within our intentions to customize the website a little further depending on the type of the user.

- For the real estate agent:
  - We will develop an environment in which he can generate letters, announcements, and so on.
  - Provide him with his own personal customized webpage
- For the estate seeker:
  - Implement a more sophisticated search form.
  - Provide the ability to post their own intent to buy or rent a property so they would be contacted about a property if it matches their criteria.

8.2 long term development

After the website is started in Sweden, the next approach would be to duplicate it on other countries. And for that, a careful study of the target countries real estate market needs to be undertaken. So the overall technical improvements would be:

- Add a country selection page.
- Add more languages
- Add payment method in case we want to make the website payable.

9. Conclusion

The real work is barely just beginning. We aim to establish a brand for ourselves, and for that we have still a long way to go. It may seem like an overly motivated project, but this is not discouraging us. Rather, we just know that we have to work at it harder. At the very least, we would have gained highly valuable experience. So far, it took a lot of thinking, learning and work to get where we are now in the project. Being the sole web engineer and designer for the website, sometimes, things weren't so trivial. But remembering the reason behind that work somehow makes it all easier. In fact, we successfully answered to the questions stated in the introduction, and addressed the needs proposed in our statement of purpose. Our website does now have:

- An attractive design
- Features allowing for the ease of use
- Interactivity with the user.
- A modern feel and look.
- Possibility to expand and scale easily

Perhaps the information and utilities we provide are not exhaustive yet, but the website is fully operable, and meant to grow mainly outside of the scope of this thesis project. So overall, we hope everyone will be using it soon to look for a place to stay.

10. References

[1] Steve Kozyk. *What is ASP.NET? -Top 12 Advantages of ASP.NET*
Available at: http://ezinearticles.com/?id=1358877

[2] www.w3schools.com. *ASP.NET vs. ASP*
Available at: http://www.w3schools.com/aspnet/aspnet_vsasp.asp

[3] Stu Nicholls (2010). *CSS3 - The future now - 'Image Slide' menu*
Available at: http://www.cssplay.co.uk/menus/css3-image-slide.html

[4] News.netcraft.com (2011). *May 2011 Web Server Survey.*
Available at: http://news.netcraft.com/archives/2011/05/02/may-2011-web-server-survey.html

[5] Commentcamarche.net(2008). *ASP - Active Server Pages.*
Available at: http://www.commentcamarche.net/contents/asp/aspintro.php3

[6] Kevin Yank*(2001). Server Side Essentials, Which Server-Side Language Is Right For You?*
Available at: http://blogs.sitepoint.com/server-side-language-right/

[7] Maui Web Site Design *(2010). Image Menu with Jquery*
Available at: http://www.alohatechsupport.net/webdesignmaui/maui-web-site-design/create_image_menu_with_jquery.html

[8] Impressive Web(2011). *CSS3 Click Chart*
Available at: http://www.impressivewebs.com/css3-click-chart/

[9] Wikipedia (2011). *Web engineering*
Available at: http://en.wikipedia.org/wiki/Web_engineering

[10] Google.com (2010). *Outils et ressources*
Available at: http://translate.google.com/translate_tools

[11] Colorpowered.com (2011). *ColorBox.*
Available at: http://colorpowered.com/colorbox/

[12] Masonry.desandro.com. *jQuery Masonry*
Available at: http://masonry.desandro.com/

[13] Oracle.com. *JavaServer Faces Technology*
Available at: http://www.oracle.com/technetwork/java/javaee/javaserverfaces-139869.html

[14] Wikipedia (2011). *Server Side Scripting*
Available at: http://en.wikipedia.org/wiki/Server-side_scripting