Polish-Japanese Institute of Information Technology Chair of Software Engineering

Master of Science Thesis

Creating a business-oriented Web portal on example of the MoMo cars online.

by

Lukasz Lazewski

Supervisor: Mariusz Trzaska Ph. D.

Warsaw, 2008

Abstract

This thesis discusses details on a portal development using the latest technologies, principles, design patterns and ideas. It attempts to solve issues of low accessibility of car sales aggregator pages using the Irish market as an example. However the author believes the document content is reasonably generic and that the principles herein would apply to many different web systems.
Ideally it could help to make some tools and technologies more popular in the industry and also help to avoid some common pitfalls that each developer makes during web design and implementation. The document begins with an overview. After that current mistakes, issues and annoyances are described together with examples. Then next few sections go on to explain the author's ideas in detail on how things can be improved together with some new unseen ideas and innovations in relation to car portal web system. This is followed by implementation details of the MoMo car web portal. The final part of the document contains the summary, pros and cons of various solutions as well as deployment hints.

Acknowledgments

To Kristyan Connolly for his support and language checking.

Adam Siemion for his positive distractions that helped me keep my sanity.

Table of Contents

1. Introduction

Few words of introduction and about context of the document.
Quick description of tools and next few chapters with achieved results.
Introduction contains four sections:

- Work Context
- Tools, systems, methodologies
- Achieved results
- About this document and its organization

Each covers briefly the document's focus and gives reader general idea what to expect in further chapters.

1.1 Work Context

The author faced frustrations of buying cars through some of the websites himself.
This work intends to show how things could be improved in the car websites and possibly change the way people design and code web applications in general.

1.1.1 Cars web portals nowadays

Currently on both the Irish and UK markets we can find many different portals offering aggregated cars sales. However there are nearly no differences between them, each offers simple drop downs based search menus to select make, model and year.
Going deeper we find so called advanced search which leads to more drop downs and text fields where the user can define their criteria.
It is not uncommon for sites to allow people to specify search parameters, which produce empty search result sets!

Set of standard available features in car portals:

- Simple search - make, model & year
- Advanced search - all other possible criteria like engine size, location etc.

Common mistakes and issues that can be observed in car websites:

- Drop downs let specify car types, which are not in the system - returns empty search.
- As stock grows it becomes harder to dig out the right car from all the search results.
- For the same reason it becomes harder for seller to sell his/her car.

The author believes that current search engines implementations for car websites are very limited in scope and are not capable of producing search result sets beyond very simple search criteria based results.
The idea behind MoMo from the beginning was to solve all the above problems and add some innovative features that would make car selling and buying a much more pleasant experience for every party involved in the process.

1.1.2 MoMo - few words about the name

MoMo name is a short two syllabic word with only two letters used to construct the word. All the author's experience with the web shows that greater than 95% of all web users are not capable of distinguishing between the Internet and a web browser.
Therefore making people type complex names into their browser or even worse forcing them to remember them is pointless and often leads to users not being able to access the website.
Most of the biggest portals on the Internet use a simple to remember or catchy name that is often not even related to their content but works well because people remember it.

Examples:

- Youtube.com - World's famous movies sharing site.
- Daft.ie - The biggest Irish house letting/renting/selling website.
- Emol.org - Aggregator about online magazines.
- Netflix.com - Movies renting website.
- Flickr.com - The world's largest photos social network.
- Bebo - The world's third social network and Ireland's most popular website.
- Delicious - The online bookmarking service

In the online car selling business in Ireland names are trying to indicate they are related to cars which has lead to many sites having either confusing or similar names which makes it difficult for them to stand out from the rest irrelevant of the quality of the site:

- autotrader.ie
- carzone.ie
- irishcartrades.com
- carhost.ie
- funkymotors.ie
- carsireland.ie
- carlist.ie

The average Irish Internet user in my research can't name more than one or two of them out off the top of his head. However everyone knows youtube and daft!
MoMo sounds like simple to remember name and it is what little kids say when they try to name vehicles so everyone should pick up easily this is car related after all.

1.1.3 Aim of MoMo as a web 2.0 cars wortal

All the above-mentioned car sites appear to be following a common pattern in regard of implementation of searching.
Neither larger websites like *carzone.ie* nor smaller sites like *funkymotors.ie* are trying to change this pattern.
Yet each and every one of them claims to be the "No. 1 car selling website in Ireland!".
How can any of them be number one if there is literally no difference except of graphics and layout? The author of this project believes that a lot of websites designing studios and web apps building companies still live in early 90s and lack new innovative ideas.

Nowadays there is a huge amount of tools and patterns available which let web developers build more user friendly and more sophisticated yet easy to use sites. The whole web 2.0 movements and ideas were just a buzzword and marketing term to start selling some prepackaged java script widgets and solutions. However some of the web 2.0 ideas bring new quality to old WWW world.

According to source materials [2] web 2.0 is aiming to:
"Enhance creativity, Information sharing and most notably, collaboration between users."

The author's personal idea about web 2.0 is that it defines new ways of presenting ideas and data to the end-user. According to the old and well-known rule that an image is worth a thousand words and that simplicity is often better than overcooked applications with sophisticated solutions that nobody uses.

MoMo attempts to implement some of those new ideas into the stubborn world of car websites.
This is achieved by adding a tag based search engine, AJAX and RSS to make cars management a breeze compared to desktop clients of current sites without their synchronization and stability problems. MoMo attempts to simplify the user interface to the extent that people are presented with all they need and no more than is necessary. Still leaves old and well known simple & advance search but also allows users to define they very own search criteria through simple text field as well as providing a tag cloud browsing system.

The web changes rapidly every day, with new technologies emerging from both big corporations like Adobe (FLEX, ADOBE AIR) or smaller projects like SproutCore which is recently getting a lot of buzz over the internet since Apple corporation deployed their new "*Me.com*" service almost entirely on SproutCore.

Some people believe this is Web 3.0 or sometimes they refer to those changes as web 2.5 – so something more advanced and sophisticated than current web 2.0. However the author believes that all of these technologies are not opening new doors. They try to implement same ideas into some other areas or in different ways with same outcome. When we look that way, sprout core is fantastic browser full javascript stack with MVC support in it. FLEX becomes yet another framework for designing and creating user interfaces quickly and by using Adobe's flash. In the author's opinion Web 3.0 will probably be more focused on distributed web service based systems.

We need to remember that each company tries to create products that could get some audience and produce some income to the producers. Therefore marketing plays big role in the way people percept the IT world and technologies, especially when we could sell same product with different name twice.

1.2. Tools, systems, methodologies

Details about operating system, tools and configurations together with short descriptions and reasoning behind their usage.

1.2.1 Systems characterization

During the MoMo development process the author used MAC OS X Leopard as a platform of choice. The main reasons were as follows:

- Automated backup system for data safety
- Reliability and stability of OSX family operating systems
- Tools available for this platform like text editors and MySQL administration tools
- Languages included in OS (Both ruby and MVC Rails framework comes with Leopard)
- Column edition for better migrations and data injection
- Development environment similarity to production deployment

Many additional tools used for MoMo development are exclusive to OSX:

- iWorks / MS Word - word processor for thesis assembly.
- Railroad – diagram generator tool (renders SVG from models in rails)
- Graphviz – Open source library and tools set for manipulating SVG graphics
- OmniGraffle - UML diagrams drawing tool.

1.2.2 Tools & Technologies description

After researching different text editors and discovering TextMate the author found this is the most powerful text editor for developing rails applications.
With its build in code manipulation and quick class switching it allows to develop applications quickly. Below are some additional features that had important influence on choosing TM over other text editors or IDEs:

- Visual bookmarks let jump through multiple parts of files
- Quick navigation and overview functionality
- Minimum amount of key strokes required for certain actions
- Full support for SVN and deployment scripts (Capistrano)

The SVN was used as the source control system of choice, main reasons were the fact that SVN versions whole revision with single version number even when just one file changes.
It's easier to obtain certain versions or changes from the past without worrying about single files as check in comments are valid to whole code base. It is also proved very reliable source management system.

For storing data author used MySQL database altogether with MySQL Administrator and MySQL Query Browser tools.

- MySQL Administrator - Allows easy management of users, databases and tables
- MySQL Query Browser - Executes SQL statements and lets find possible optimizations for defined queries.

In addition following technologies are part of MoMo:

- Ruby - fully object oriented language with multiple libraries available for variety of tasks.
- Ruby on Rails - Full MVC stack implementation in Ruby language.
- ImageMagick - freely available written in C images manipulation library.
- Mongrel - application server used to process all dynamic queries coming into MoMo.
- Prototype - unique and very stable java script framework for DOM elements manipulation it comes as a part of Rails framework.
- Nginx - Russian made reverse proxy server with the smallest memory footprint and high responsiveness.

1.2.3 Methodologies for work organization

Due to limited time and the fact this work is highly dependent on three different customer groups (buyers, single time sellers, car dealers) the author had to find an optimal balance between acquiring information, system design and coding.
As a result of the mentioned dependencies the project was developed with agile principals that include extreme programming that allowed to work close and respond quickly to ideas and suggestions made by car dealers and end users.
This allowed the author to obtain valuable information about what current systems are doing wrong and what could be improved and what is currently missing in existing systems.
Implementation of their ideas and opinions into MoMo users interface and backend allowed the project to grow into what it is now.
With agile development model in mind some changes were dynamically applied and the results presented to mentioned group of participants for their feedback.

1.3. Achieved results

Achieved results in MoMo are innovative and will hopefully increase number of developers following standards and using some of the tools the author experienced during his work.

1.3.1 What MoMo has

MoMo delivers all well-known and commonly used search functionality, which include simple make & model search plus advanced search and browse.

In addition to these a completely innovative search engine for finding cars was developed.
MoMo is the first car web based system to implement a tag based search engine.
Which provides a more powerful and user-friendly search experience. It allows users to search on many different car features and aids them by providing an auto suggesting text field, which helps them, select tags easily. MoMo also provides the most popular tags in a form of multiple tag clouds throughout the system. Additionally dealers may trade cars among themselves this is known as MoMo’s in trade market.
MoMo as one of not too many systems runs completely on web-server therefore there is no problems with the licensing, versioning, synchronization of stock differences or installation

issues on different machines and OSes.

1.3.2 Car sellers feedback

As mentioned previously different people involved into car selling process had huge impact on what MoMo has to offer and the way it does things. Involved people were able to point out features they wanted and features they found completely useless. This allowed focus development on certain areas leaving others for later or dropping them completely.
The author learned about all the frustrations and issues of current car sales aggregating sites, which are:

- Poor support and assumption there is no way of doing things better from sites operation teams and owners.
- Most of them tend to depend on thick client for cars upload which introduces synchronization problems. Author had chance to look into some clients and they were very unstable. Users were advised to restart they machines constantly in case of any trouble.
- One instance of thick client - dealers were not allowed to install it on more than one machine.

1.3.3 MoMo market deployment

Currently MoMo is not publicly available (august 2008), however it is intended to change shortly.
Engine Yard hosting company founded by **Ezra Zygmuntowicz**[1] would be first choice for MoMo's deployment.
Ezra - Industry famous person has very good experience and past with different deployment schemas for rails. Engine Yard Company should let MoMo start small and grow while it attracts more users. Ideally MoMo will use multiple mongrel app servers in cluster and Nginx to distribute load between them and deliver static content such as images, java script files or style sheets. The MySQL server would be running on separate physical box accessed by local network and with proper redundancy policies in place.

1.4. About this document and it's organization

Text is written with British English pronunciation and orthographic rules of typing words, for example: *colour* instead of American English *color*.
Some smaller images are shown directly in text while big ones often whole page in size can be found in attached images reference sheet. References to external sources are written in rectangular brackets example: [1]. Please refer to references page where they are described in detail.

This document is not only containing information about MoMo itself but as well additional

1 http://www.bestechvideos.com/2008/07/05/ezra-zygmuntowicz-on-engine-yard-and-rails-deployment - Ezra Zygmuntowicz has worked in PHP and hand-blown glass art, and now uses Ruby for web application and system automation programming. His work as the webmaster for the Yakima Herald-Republic newspaper taught him a lot about Rails Deployment architecture and how to scale a Rails app. His book on Rails Deployment is due from Pragmatic Programmers in June 2007

information regarding technologies and solutions which can be applied to many different dev situations, scenarios or projects. Author's aim was to show variety of system's aspects, which includes possible algorithmic and coding solution examples, set of tools, ideas or standards for better web portals development.

Text is organized in 6 chapters with such content:

Chapter 1 - Contains brief information about MoMo and car market.
Chapter 2 - Describes current status of cars oriented websites with all their pros and cons.
Chapter 3 - Delivers information about tools, methodologies and technologies used during MoMo development.
Chapter 4 - Gives the detailed tour of MoMo's features.
Chapter 5 - Shows system's architecture and implementation of key elements.
Chapter 6 - Summarizes the outcome of work around MoMo.

In addition to these thesis contains following appendixes:

Appendix A: Dictionary
Appendix B: References & Literature,
Appendix C: Schemas and big Images

2. Currently Available Car Sites

This chapter is showing currently available car sites on the Irish market together with pointing out some issues and presenting case studies on the most popular ones.
Defines every single aspect of pros and cons altogether with suggestion how to solve current downsides and issues.

2.1. Definition of current portals

In this chapter the author tries to show current portals and their issues on an example of the Irish car market sites. All mentioned problems are not uncommon to other websites and could be still seen elsewhere.

2.1.1 Too heavy, overloaded with information and chaotic

Unfortunately the web evolved in very random and spontaneous way where small group of people tried to create some standards and only a small percentage of web developers tried to follow them or understood them. Internet is full of websites where arriving for the first time viewer has an impression that they were created by coders for coders, without straightforward interface with unnecessary features directly on the main site. People try to put as much as they can so they portal look big, important and professional, sadly results are reverse and the audience is who suffer.

Its not about proving to the world how many technologies, tricks and widgets could be packed onto website, most people want just basic features. It is proven that some sophisticated functionality on some sites is used by more advanced users mostly from IT industry – this is only some small chunk of percent of all visitors.

For good this is changing – modern frameworks and people who are responsible for the technology progress are leading the way and teach others how things should be done or simply limit mistakes. Such examples are modern limits in frameworks by disallowing mixing logic with presentation layers (MVC pattern in Ruby on Rails).

2.1.2 Standard two and three columns layouts, lack of innovation, hard navigation

Every developer knows three and two columns layout, they are pretty basic easy to achieve and often follow *golden rule* of 60-40 ratio which is said to be easy to read.
As much as it is true all websites are trying to follow that, this is boring and portals are all the same in shape. Web 2.0 introduces few good ideas, not only innovative but often revolutionary:

- Make website's logo stand out so it is obvious what people are looking at.
- It is good to use 'badges' some quick information may be outlined with absolutely positioned graphical element that should look like promotion messages in shopping malls. They certainly grab attention.

- Use big fonts – H1, H2 elements are now important from spiders point of view so do not be afraid to use them. Besides bigger font should mean more important things to read.

- Separate header and footer from the main site, make sure site is visually separate on each.

- Keep most important things in menu, but just limit them to couple of links so people can easily find what they are looking for.

- Try to visualize data rather than present bare text. Guarantees better readability.

Most of them are what nowadays web 2.0 sites have and it seems like this new trend in websites' layouts is getting attention and achieving huge success among everyday not specialist web browsing users. [4]

2.1.3 Irish market and car sites

In this thesis author focuses on car sites on the Irish market. Idea comes as author lives and works in Ireland for three years now and his personal experience with finding car at Green Island were very negative and frustrating. As mentioned in earlier sections all problems apply for car sites too. The main disadvantage about them would be overwhelming amount of information site owners are trying to present to both buyers and sellers.
In addition some of them still use thick clients to upload stock of dealers, this is asking for synchronization and versioning of software problems. They use some old desktop technology – tied to windows platform and very unstable.
Quick look at the urls structure shows that all data is passed as get parameters, further investigation with HTML code check shows that most forms are not making POST requests either so the whole application works through GET-s. It is ok as long as owners agree to easily give out all the details about parameters and values that server accepts.
When modifying any of them with big values as expected database dumps the whole content. Not only insecure but as well inefficient – possibly reason why sites are very slow during peak hours. For legal reasons links breaking these applications won't be presented in this document.
Another thing is their ways of doing business. Average person who wants to sale a car has to pay thirty euro just for putting their Ad for couple of days.
In era where everybody finds it hard to gather content to the website and most of it is free it is at least surprising. Compared to eBay where anyone could start selling anything for free and company gets only some small income at the end of transaction.

2.1.4 Case studies - Carzone, CBG and AutoTrader.

There are really only few sites related to the car market in Ireland that stand out off the crowd and try to do things differently. Carzone is one of them, actually what author learned is that Carzone was one of the first on this market and is at the very top since then.
Carzone's website was update just few months ago, before it was early 90s style like website with two search boxes and nothing major. When work on the MoMo was starting it was nowhere close to what it looks now. There is huge progress on the UI side but still it only does few things like simple and advanced searches.

What's more many people believe that information about insurance and all of presented statistics is not professional and unnecessary only cluttering the page and bringing confusion. Fortunate for Carzone big distinction of actual search boxes and car sale ad makes it visible so average person would know steps to sell or buy a vehicle.

Another big car site would be CBG – Cars Buyers Guide. This website used to win many awards for its design, nowadays however it is full of badly used AJAX calls which are making website slow and sometimes let people find more vehicles than number indicates or opposite. Yet again this car site does not try to show off anything new except for well-known drop downs. Third example of big car portal in Ireland is the Autotrader – similar to CBG, Autotrader comes from car journals background.
However Autotrader is actually a big global company with millions of cars in stock all over the world.
Interestingly their website in Ireland is not that cluttered and pretty easy to understand.
Even the basic layout with outdated graphics and still standard functionality makes it quite popular among Irish car buyers and sellers.

2.2. Issues to solve in car websites

When the car system grows it becomes harder to find certain vehicles as pagination shields users from all the possible results. Even the latest add-ons such as keyword searches have their limits and their current implementation makes them useless in most cases.

2.2.1 Finding proper vehicle when market grows too much

Many car dealers in Ireland believe that after a certain number of cars is in stock it becomes very hard for them to sell their cars because people go through up to three pages of search results and pick vehicle from there. So when there are many more cars in the system they may not be sold or it may take long time. Certainly this is bad for buyers too as sometimes better vehicles come last in search.
Reasons behind it are bad sorting and number of results per page, however no matter what there always will be some cars that are after page third. CBG tries to solve that problem by letting sellers pay additional fee for showing their cars first. That solved the problem temporarily, as nowadays no dealer would mind to pay additional small fee to get his cars into this "special" list – so all of them are now special and we are back at the beginning of the problem.

2.2.2 Keyword searches limitations

Some sites offer keyword searches. For variety of reasons they are very limited and do not allow finding what we may intent to. For example when typing: "blue" and with blue car in mind we would still get other colours of the cars – how is that possible?
Keyword search is performed on every piece of car's description if there is "blue" word anywhere across car ad it will be returned. So all cars from "*Bluelagoon car sales*" dealer apply. The same if address, model or any other field contains word blue.

2.2.3 Data presentation

All of earlier mentioned sites are displaying search results as HTML table where rows are showing data one by one. None of them tries to show the most of the car with only few showing thumbnails in search results. While looking at the car ad itself there is no way to find similar vehicles by different categories, potential buyer may only see search results again.

No similar vehicles are shown and could be navigated to by simply clicking on them.
All sites' creators assume that back button in the browse is to be used for. Ideas like breadcrumbs that notify about status and position in portal aren't present.

2.2.4 Can any of this be done better?

See in next chapter how MoMo implements browsing and tagging to let people find many cars through different ways and how RSS actually let people observe stock changes live.
MoMo's search results try to solve the problem of readability and showing most of the vehicle. M-tag search lets build custom queries based on not one but multiple keywords/tags that return exactly what people intend to get.

3. Tools, methodologies and technologies used during MoMo development.

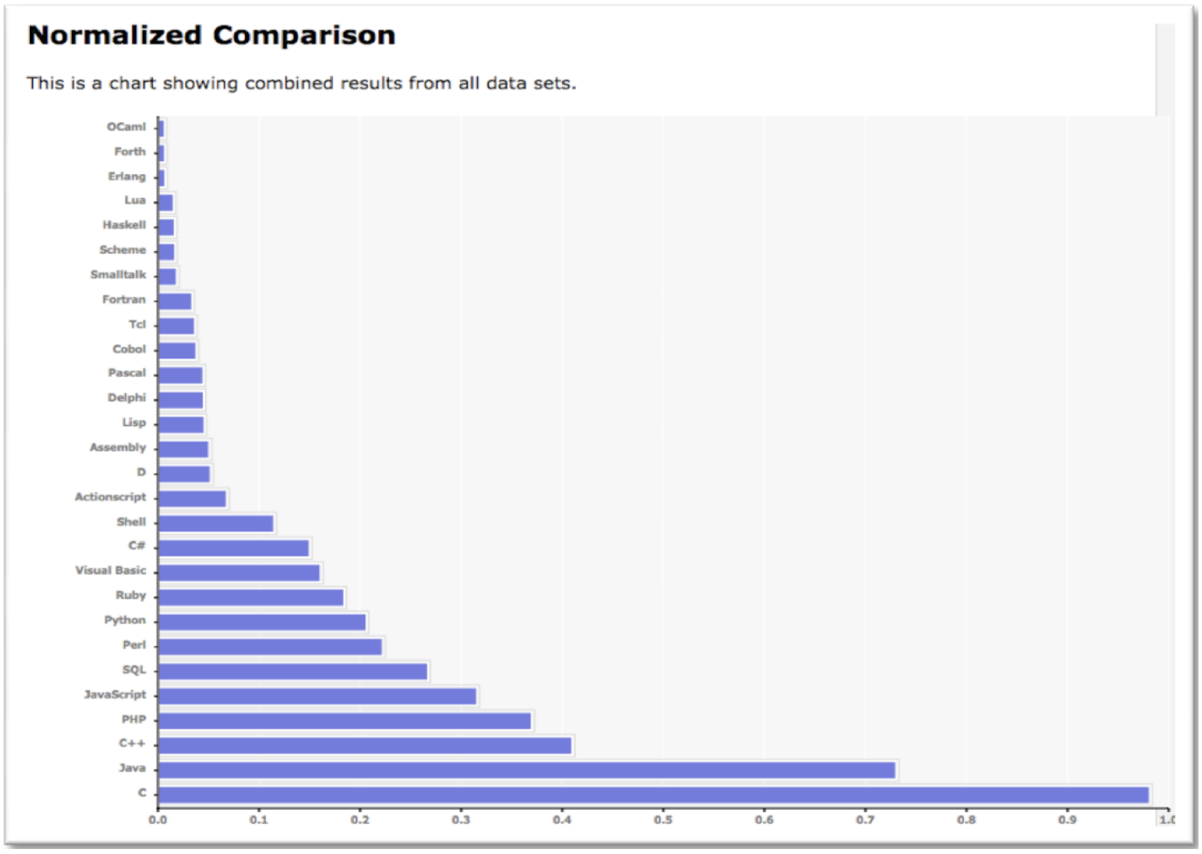
Nowadays number of tools for developing desktop or web applications is enormous and hard to choose from. Purpose of this chapter is to show some common, easy and popular tools that are very well tested and stable. Their usage is not only limited to such a products as MoMo but generic and could be applied into many other development situation.
Additionally a few new technologies and techniques are presented and hopefully this chapter will make them more popular for development teams and amongst people making decisions.

3.1. Technologies

This section is about the technologies and tools used during MoMo's development.
All described here languages, frameworks and tools can be applied to any other web project.

3.1.1 Ruby as a language of choice for creating MoMo

Ruby is fully object oriented interpreted language. Created by Yukihiro Matsumoto from whom existing languages were not natural enough. His idea was to connect power of Perl and Python with object-oriented Smalltalk and to achieve straightforward and natural language that could map thoughts into the code. Outcome is very flexible and nowadays popular language – Ruby. According to *http://www.langpop.com/* that gathers data from different sources and presents aggregated information (Image 1) Ruby managed to get some serious attention from developers.



(Image 1) Languages popularity aggregated data according to www.langpop.com

Interesting idea with ruby is based on popular in Linux world software packaging and delivery system.
Similar to apt-get tool in Debian Linux distribution GEMs are set of scripts written in Ruby that allow easily find any library together with their dependencies.
After that they perform installation automatically by downloading all necessary components from the Internet.
For that purpose GEMs have special repositories where third party developers may upload their packages for sharing.
The system described in this thesis uses variety of GEMs for many different tasks including image compression or different view template language.

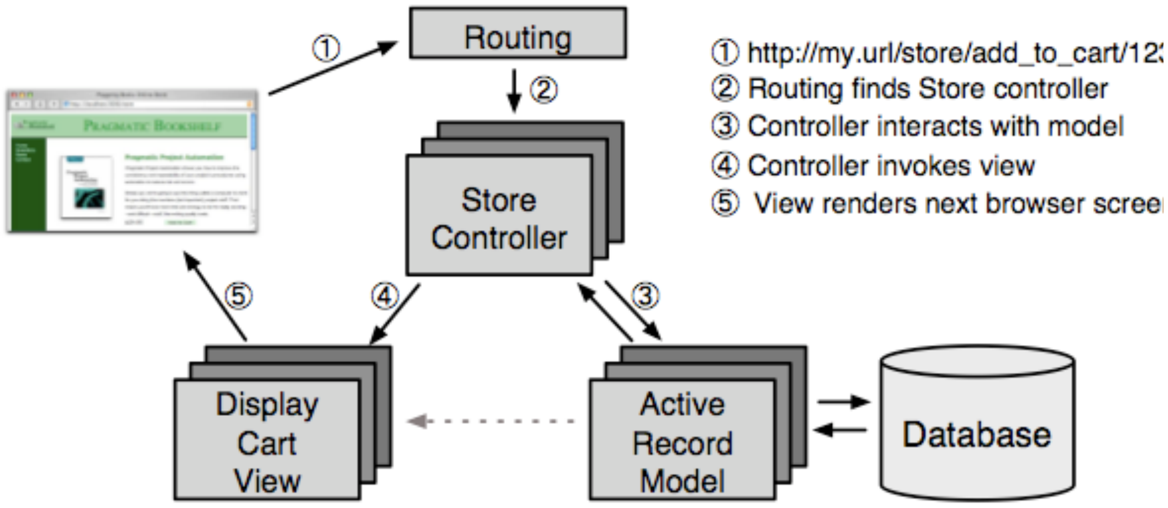
3.1.2 Ruby on Rails as MVC stack

Every modern framework comes as someone's idea to solve some specific problem, neither of them is universal enough to solve all possible issues.
JAVA world is very good example, to build web application with language of this magnitude we often need to define lots of XML files which are different configuration files and deployment descriptors.
Ruby on Rails (later Rails or RoR) was designed and created by David Heinemeier Hansson as a lightweight yet, powerful framework build on ruby language. Whole idea was based on ActiveRecord which is Object Relational Mapper - just one of the RoR's components.
Later as project have grown additional plugins and projects got into the core of what Ruby on Rails is today.

David did great job with figuring out few ideas that should be followed by all other major companies, communities and developers. He redefined thinking about web, here author attaches just few simple rules Rails's creator introduced:

- DRY – do not repeat yourself
- Keep things where they belong – MVC framework guarantees separation between database access layer, controllers and users' interface.
- Make coding simple and fun – Rails and ruby language in general has very low entry level for developers to pick up.
- Overdo through under doing – often people throw tones of features making their applications overwhelmed with buttons and widgets following fake impression that more is better. People accessing their products are confused, lost and quickly navigate away from their sites.

Rails maintain routing in *routing.rb* file where different url to action mappings are formulated. However in opposition to JAVA or .NET developers may map resources – idea directly coming from Restful principles [rel. 2] with CRUD mapped to urls through HTTP protocol actions. *Image 2* shows flow of rails application.


Routing
Store Controller
Display Cart View
Active Record Model
Database
① http://my.url/store/add_to_cart/12
② Routing finds Store controller
③ Controller interacts with model
④ Controller invokes view
⑤ View renders next browser scree

(Image 2) Application Flow in rails [5]

Rails supports so called environments, which are predefined types of deployment for applications. With this in mind RoR core team defined:

- Development – all files are always preloaded per request basis so there is no need to redeploy, restart or interact with application servers in any way.
- Production – all caches are on, application compresses most views and rails on first request are interpreting them, after that for most time they are just preloaded from RAM.

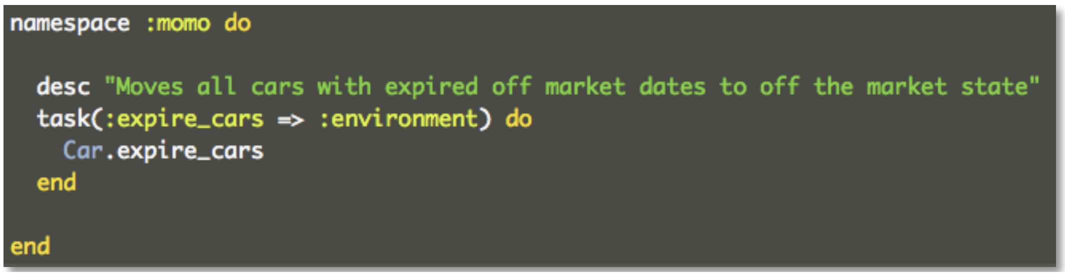
Logger runs in info mode so no extensive information will be outputted into logs.

- Staging – similar to Production with more extensive logging.
  It has exact copy of Production's database to refer to real live data in separate physical DB instance.

- Testing – Merge between production and development where we can see all output from logger and MySQL logs. Usually has dummy data worthless and not causing any casualties when lost.

3.1.3 Rake the Ruby Make tool

Rake is equivalent of make tool in other languages, could be compared to JAVA's ant or maven. Developers type tasks in ruby itself, no need to write XML or worry about indents as in standard makefiles. Rake comes as GEM and contains predefined libraries and tasks for most of common operations. Rake supports all rails components like ActiveRecord for databases operations. Please consult *Image 3* to see rake snippet used in MoMo for expiring cars on market depending on the environment.



```
namespace :momo do

  desc "Moves all cars with expired off market dates to off the market state"
  task(:expire_cars => :environment) do
    Car.expire_cars
  end

end
```

(Image 3) Rake task for expiring cars

3.1.4 Prototype, Rails' javascript framework of choice.

Prototype is one of the most mature java scripts frameworks available for developers nowadays. It shares some Object-oriented principles where some certain logic is encapsulated in objects and objects then gathered inside Libraries.
As an example AJAX library is containing all remote and asynchronous calls functions including *Ajax.Request* and *Ajax.Update*, which are commonly used in MoMo.
Another thing is that Prototype makes all DOM elements on the page manipulation straightforward and easy. So removing rows of tables without reloading whole page is a matter of having unique DIV ID for each row and call *Page.remove(div id).* Again this is nicely wrapped inside Page module of Prototype. Last but not least is the fact that prototype seems completely browser independent. Big win in world where all different browsers parse HTML differently and some of them do not follow any standards.

Rails developers created many handy helpers for generating prototype calls directly inside views. They come bundles with rails and theoretically let people do not type single line of javascript as everything can be done by these helpers.

Examples include Ajax Remote links calls:
link_to_remote "Click Me", :url => { :controller => "cars", :action => "display" }

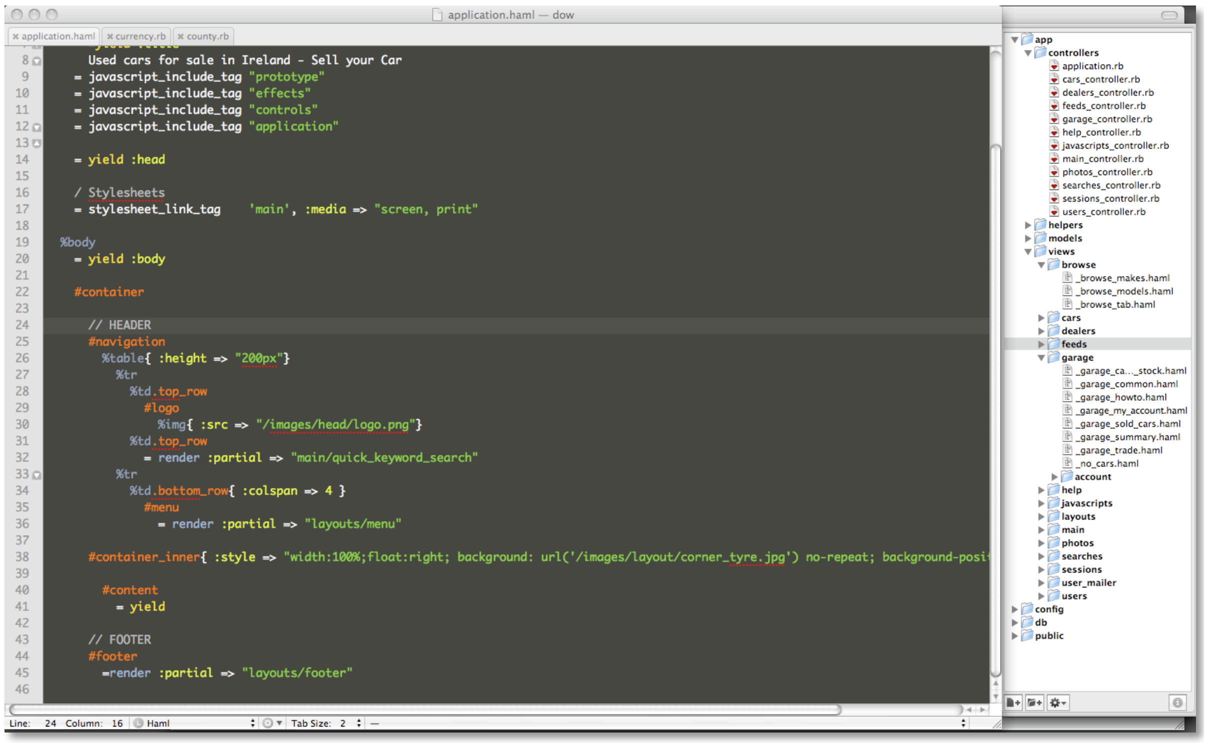
Rendering:
new Ajax.Request('/cars/display ', { asynchronous:true, evalScripts:true });

From developers point of view making calls like above are simpler when handled by one type of logic, so views are all written in ruby without snippets of JS around.

3.2. Design, diagrams and database creation tools

Section presents different tools altogether with reasoning behind their usage and screenshots. Links to tools' websites are in Bibliography.

3.2.1 TextMate ultimate text editor for rails



```
      Used cars for sale in Ireland - Sell your Car
    = javascript_include_tag "prototype"
    = javascript_include_tag "effects"
    = javascript_include_tag "controls"
    = javascript_include_tag "application"

    = yield :head

    / Stylesheets
    = stylesheet_link_tag    'main', :media => "screen, print"

  %body
    = yield :body

    #container

      // HEADER
      #navigation
        %table{ :height => "200px"}
          %tr
            %td.top_row
              #logo
                %img{ :src => "/images/head/logo.png"}
            %td.top_row
              = render :partial => "main/quick_keyword_search"
          %tr
            %td.bottom_row{ :colspan => 4 }
              #menu
                = render :partial => "layouts/menu"

      #container_inner{ :style => "width:100%;float:right; background: url('/images/layout/corner_tyre.jpg') no-repeat; background-posi

        #content
          = yield

      // FOOTER
      #footer
        =render :partial => "layouts/footer"
```

(Image 5) Textmate in action

TextMate from Macromates become very popular since RoR author has pointed it out.
This editor fully deserves current attention. It connects simplicity with lightweight interface and few very powerful features like automatic search of files with easy access through keys combination. Allows finding corresponding files for controller its views and used models.
TextMate is fully scriptable so we may want to define some common options into set of scripts, which are then, bundled together for easier sharing with others.
For this reason there is extension for majority of programming languages and supportive tools like column edition, currencies, variety of build tool, system commands, html/xml tags, source control tools like CVS, SVN and recently even GIT.

Each major language has code highlighted and properly resolved all the variables, data types and functions/methods. Additionally there are no limits to look and feel anyone could define their own schema for displaying any particular language.
With Unix principles behind it people can easily add their own features, it is more framework like tool where people could shape it the any way they want.

3.2.2 Mysql tools - databases management, tuning and querying software

For systems of MoMo's size it is obvious to choose from one of available databases systems, MySQL is the most popular, well tested, open source and most important freely available.

No DB however is useful without good set of tools for managing, maintaining or testing and optimizing queries. MySQL founders offer a couple of very good and again free tools for this purpose.
MySQL Administrator grants developers quick access to:

- Quick database setup (creation, update)
- Accounts management – users creations and their privileges management
- Setup data redundancy policies, backups and recovery
- Multiplatform – works on Windows, Linux and other Unixes including MAC OS X
- Health monitor – shows queries per second, problems such as memory usage peaks and logged data base information.
- Server Optimization– provides choices of different DB engine, tuning them and set caches.

MySQL Query accesses databases' tables and their rows for viewing and editing data.
It's a very good tool for testing defined queries like store procedures or improving them based on type of data and columns they return. Some times it isn't possible to just type in ORM based query and see what comes back. Developers should always try to test their SQL no matter if it comes from ActiveRecord or is written by hand it may always do things little bit different then intended or in inefficient manner.

3.2.3 MAC OS X Leopard - platform of choice

The author always had very good experience with the Unixes like systems including different flavours of Linux. Some software however is only available for commercial platforms where FreeBSD and Linux don't belong. Secondly open source systems come without guarantee of any kind and all problems are to be resolved by developer himself. This usually costs time and in companies has direct correspondence to money.
Creating a web project designer should be prepared to test it in all major browsers including MS Internet Explorer, which in different versions has according to different sources about 70-80% of the market share.

This is where Leopard comes into the picture. It is sixth release of MAC OS X family operating systems by Apple Corporation. Solves most if not all problems author faced during development process.
Leopard comes with few unbeatable features like *time machine* which is build in back up software making copies of user's content every hour to the destination of choice basically eliminating any chance of loosing code.
For testing in mentioned Internet Explorer author installed windows operating system through *bootcamp.* It is another build in property of OSX - allows to have windows on any Intel CPU based MAC computer on separate partition. When booting machine we can then choose which OS we want to boot into. To extend that further by using third party software like VMware virtualization this second partition windows could be run inside Leopard as with all it pros and cons. Very same idea and rule apply for windows' applications including web browsers.
Most importantly Ruby and Ruby on Rails come bundled into OSX and mentioned earlier TextMate is only available for this platform. In general most Rails developers describe Leopard as reliable, very high performance and stable system that anyone could depend on.

3.2.4 Railroad diagram converting and drawing tool

As design in every RoR application starts at DB level it would be great to let people create their schema and then render UMLs based on that. This on first sight reverse order of doing things in software engineering is getting more and more popular, mainly because in projects of MoMo's size when single person creates whole product it is often done through hand drawings and quick notes on bits of paper. DB schema evolves and then in final form we could proceed with making UML diagram. Otherwise every time there are any changes designers would have to refresh UMLs.
This is where Railroad ruby application comes handy. It allows rendering schema after reading DB tables using ActiveRecord. This way application architect receives valid schema in matter of seconds as SVG – vector graphics output for later interpretation by whole palette of tools either open source, free or commercial.

3.2.5 iWorks / MS WORD as a word processor

When starting on MoMo the author tried to write this document in iWorks pages.
It is word processor from apple and comes as a text editor in iWork package, which is direct competitor to MS Office. It nicely integrates with operating system and is far more lightweight and easier than Word.
However pages main advantage – OS X integration is its main disadvantage too.
Other people interested in this document had no chance to put their input as pages could only render PDFs. After that decision had been made to move work to *Word* and carry on with this tool for text assembly.
Thesis is written with Word 2008 from MS OFFICE for MAC Package and nicely converts to equivalent formats for other platforms.

3.2.6 Drawings and code snippeting

To present some nice and straightforward schemas and drawings author used Keynote presentation tool from iWorks package and its build in tools for creating custom schemas.
The *Grab* has been used for presenting code snippets.

4. MoMo car wortal

This part of document introduces author’s implementation of the car portal – MoMo.
It shows off main functionality including some completely new and unseen before ideas like tagging, m-tag search and multiple per tag based RSS feeds.
Introduces interface enchantments for sellers called garage and few site navigating tricks.
At the very end describes RSS feeds for better market observation and access for dealers especially those using trade mode – special dealer-to-dealer sales only functionality.
There are no implementation details, they are covered later in section 5.

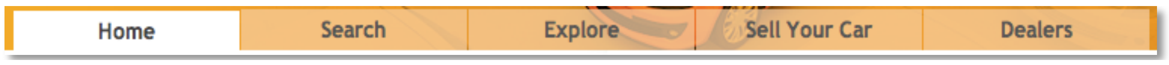
4.1. Navigating in MoMo

Different ways of navigating in MoMo are described in this part of the document.

4.1.1 Main and Head menus

Main menu in MoMo (*Image 6*) is organized with minimal amount of information in mind, trying to show only what is the most important and what user may need.
It contains following elements:

- Home - displays main page of the website. It is starting point for Memo.
- Search - contains simple search, advanced search and browse (read following section 4.2)
- Explore - contains explore functionality altogether with tag searches (check out point 4.3 of this chapter)
- Sell Your Car - contains all basic information on selling cars with MoMo plus allows people to register quickly.
- Dealers - Lets find closest dealers, as well groups them by region or shows in alphabetical order. Their profiles are being presented with their stock.



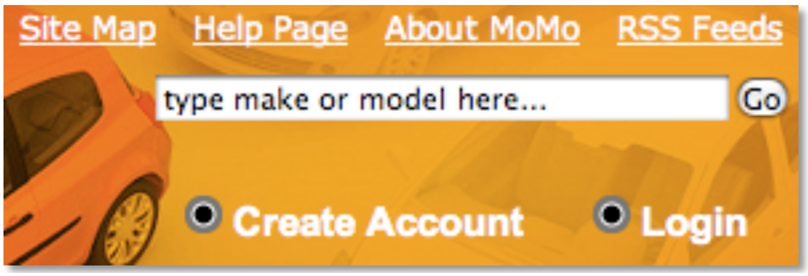
(Image 6) Main menu in MoMo

Elements are presented transparent or white when selected so a user exactly knows where they are plus nice UI effect is achieved this way too. When seller is logged in ‘Sell You Car’ is changed to ‘Garage’ which leads to part of system allowing stock management for user’s cars. (*Garage is fully described in 4.5 point of this chapter*)

In addition to main menu some lower priority information are presented in top right corner.
This menu contains quick search too. We could type make & model to quickly find some cars.
Other elements in this menu are ‘Login’, ‘Create Account’ for easy signing up, signing in or ‘Logout’ when logged in.

Site Map, Help Page, About MoMo and RSS Feeds leads to corresponding sections of the portal. They allow viewing portal’s site map, helping pages, and reading about MoMo itself and preview RSS Feeds available for subscription. (*More on RSSs can be found in 4.4 section of this chapter*)

Reader is advised to acquaint himself with *Image 7* that presents 'Head Menu'.

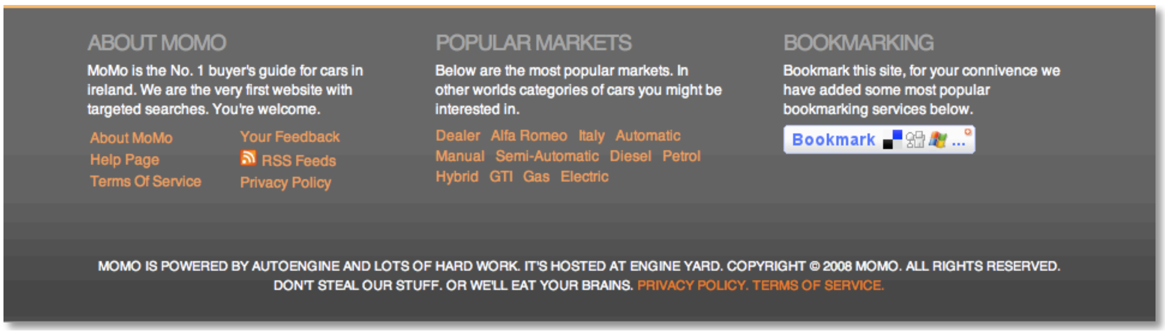


(Image 7) Head Menu in MoMo

4.1.2 The Footer

Although footer is less visible it still occurs on all pages as an important element of whole layout. With web 2.0 principals in mind author designed footer to contain some uncommonly used information. Links in footer correspond to mentioned 'Head Menu', contain some popular tags for quick access and offers bookmaking services to let people use whatever they prefer for preserving site in their service of choice.
As well privacy policies and terms of service are accessible there.

Footer element comes in three columns layout with separate heading on each so they announce what column is for. *Image 8* shows MoMo's bottom section.


ABOUT MOMO

MoMo is the No. 1 buyer's guide for cars in ireland. We are the very first website with targeted searches. You're welcome.

About MoMo
Your Feedback
Help Page
RSS Feeds
Terms Of Service
Privacy Policy

POPULAR MARKETS

Below are the most popular markets. In other worlds categories of cars you might be interested in.

Dealer Alfa Romeo Italy Automatic
Manual Semi-Automatic Diesel Petrol
Hybrid GTI Gas Electric

BOOKMARKING

Bookmark this site, for your connivence we have added some most popular bookmarking services below.

Bookmark

MOMO IS POWERED BY AUTOENGINE AND LOTS OF HARD WORK. IT'S HOSTED AT ENGINE YARD. COPYRIGHT © 2008 MOMO. ALL RIGHTS RESERVED.
DON'T STEAL OUR STUFF. OR WE'LL EAT YOUR BRAINS. PRIVACY POLICY. TERMS OF SERVICE.

(Image 8) MoMo's footer

4.1.3 The Bread Crumbs

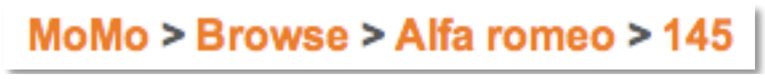
Web 2.0 era introduced couple new ideas to navigating websites, apart from Ajax calls which are rendering whole HTML chunks of code or Flash based websites which are heavy and require third party plugins to support them in a browser.
Developers may introduce less sophisticated and basic features, which assure users, will understand and use.

Breadcrumbs were well known before web 2.0 ideologists coming up with their ideas and visions of current web. Nevertheless they are having real renaissance and are used on every major website. Author found them useful to remind 'position' or history of the current opened sub-pages. Besides they let navigate back to change things quickly aiming for better overall navigation. MoMo uses breadcrumbs for nearly every page. Some are just static links

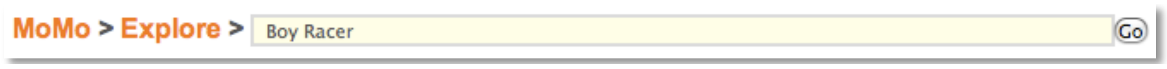
displaying pages in hierarchical way separated with slashes while others dynamically generate to reference users search parameters, tag or browse choices. They are placed in highly visible place just beneath the main menu and page's title.
Set of colours picked in style sheet guarantees their high visibility and assures they will stand out from the whole page so no one can miss them or misjudge their function.

List of possible uses of breadcrumbs with references to image representing them:

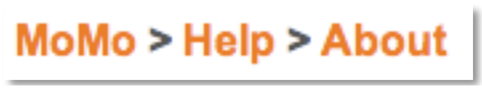
- Browse: Shows user that he is browsing and allows going back for choosing different make or current make's models.
  Image 9 shows this kind of breadcrumbs - they are being rendered automatically for each query.
- Explore breadcrumbs are very unique on their own, their last choice is put into text field allowing buyers to easily modify it directly inside breadcrumb. *Image 10* is an example of Explore breadcrumb. Its available in both search results for tag and explore tab.
- Static breadcrumbs navigate through static pages, they are being rendered only once, for the first time when server starts and page is accessed.
- Good examples are all helping pages and their subtopics. (*Image 11*)



(Image 9 Breadcrumbs on browse example)



(Image 10 Explore breadcrumbs with typing possibility)



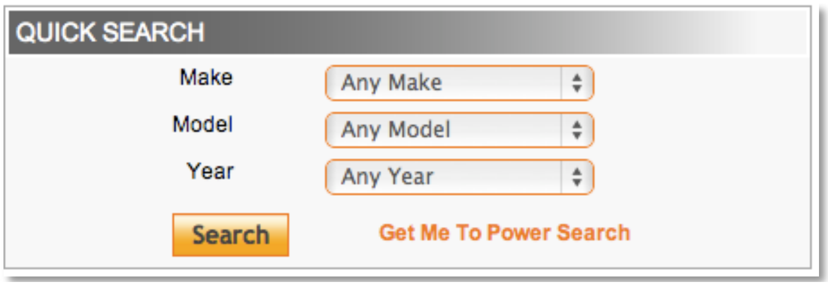
(Image 11 Static breadcrumb examples)

4.2. Searching in MoMo

MoMo offers well-known and commonly available drop downs searches as well as some completely new ideas. In this section reader can learn about all different possibilities of finding vehicles in MoMo.

4.2.1 Simple make, model & year search

As any other car web portal out there MoMo lets users to perform simple search where buyer can choose make, model and year of vehicle he or she intends to find.
This functionality is available via main page where drop downs are available at center of the page and as well very same partial with different style is used on 'Search' tab of the application. Similar result offers always available text field in top right corner of browser's window, however difference here is that we can start typing any make or model and system auto suggests possible searches we can perform.



(Image 12) Simple Search aka Quick Search

4.2.2 Advanced search

Advanced search called in MoMo Power Search is a big html form with different choices where buyer can specify every single aspect of a vehicle to find.

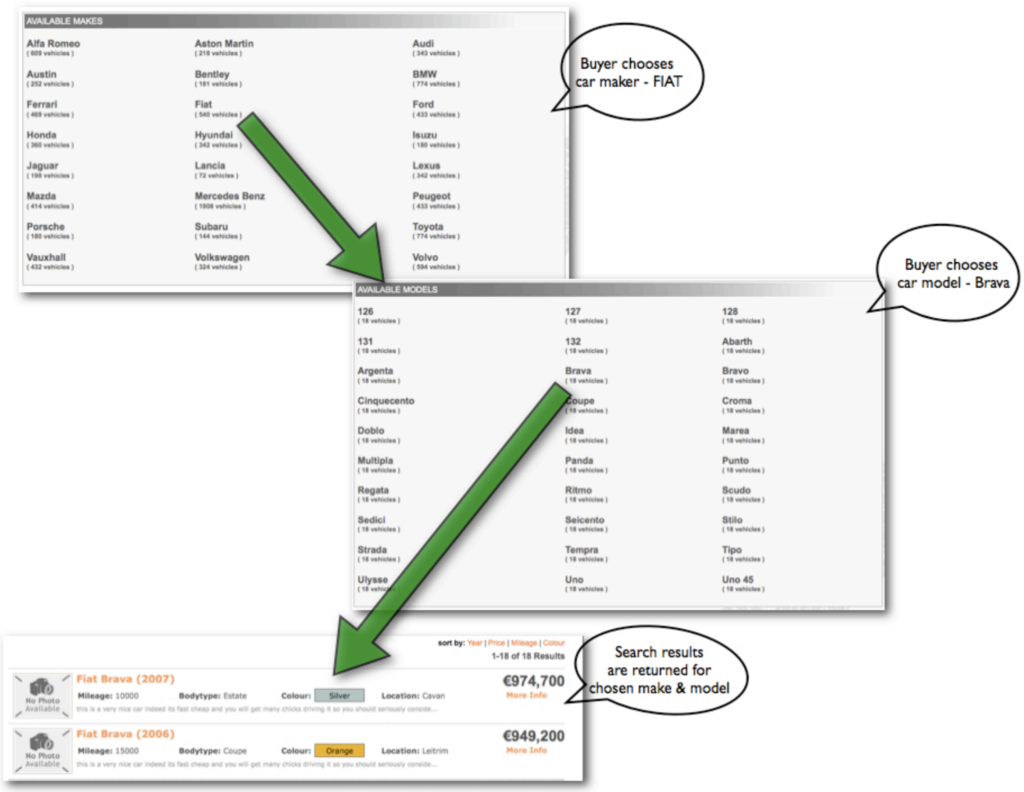
Form contains following choices, which are all self-explanatory:

- Make & Model drop downs.
- Maximum and Minimum prices range
- Maximum and Minimum mileage range
- Maximum and Minimum engine sizes range
- Maximum and Minimum years range
- Doors number
- Body type of the vehicle
- Engine type
- Colour
- Transmission - (manual, automatic, etc.)
- Condition - (new, very good, damaged, etc.)
- County - (region car seller is from)
- Dealers - (drop down lets specify exact dealer)
- Seller Type - (private or dealer)
- Photos - (with and without photos search)
- Added - decision on since when added cars we want to see.
- Sort by - sort results by, possibilities are: Year, Price, Mileage and Colour

This is quite standard advanced search form, however in opposition to other website user is not forced to pick any of those choices. (Elsewhere people are obligated to choose at least make and model) This means we can specify all vehicles to certain year, budget and county they are from. Author made sure that common sense thinking applies here, most of us like some specific brands of cars and makers therefore they will be looking for those specific ones. Still some people prefer to find vehicle which meets they budget - all they care is to how the best spent their money.

4.2.3 Browsing for cars

In addition to earlier mentioned Search and Power Search car buyers can browse for cars. On ‘Search’ Tab of the main UI people can choose one of available in MoMo carmakers.
By clicking on them user will be presented with all accessible models for that maker in database. This corresponds to real cars in the stock and is self-adjustable so people cannot browse for non-existing makes and/or models. Selecting exact model will render search results for all cars for that model. *Image 13* is presenting steps to follow for browsing experience.



AVAILABLE MAKES

Alfa Romeo (609 vehicles)	Aston Martin (218 vehicles)	Audi (343 vehicles)
Austin (252 vehicles)	Bentley (181 vehicles)	BMW (774 vehicles)
Ferrari (469 vehicles)	Fiat (540 vehicles)	Ford (433 vehicles)
Honda (360 vehicles)	Hyundai (342 vehicles)	Isuzu (180 vehicles)
Jaguar (198 vehicles)	Lancia (72 vehicles)	Lexus (342 vehicles)
Mazda (414 vehicles)	Mercedes Benz (1908 vehicles)	Peugeot (433 vehicles)
Porsche (180 vehicles)	Subaru (144 vehicles)	Toyota (774 vehicles)
Vauxhall (432 vehicles)	Volkswagen (324 vehicles)	Volvo (594 vehicles)

AVAILABLE MODELS

126 (18 vehicles)	127 (18 vehicles)	128 (18 vehicles)
131 (18 vehicles)	132 (18 vehicles)	Abarth (18 vehicles)
Argenta (18 vehicles)	Brava (18 vehicles)	Bravo (18 vehicles)
Cinquecento (18 vehicles)	Coupe (18 vehicles)	Croma (18 vehicles)
Doblo (18 vehicles)	Idea (18 vehicles)	Marea (18 vehicles)
Multipla (18 vehicles)	Panda (18 vehicles)	Punto (18 vehicles)
Regata (18 vehicles)	Ritmo (18 vehicles)	Scudo (18 vehicles)
Sedici (18 vehicles)	Seicento (18 vehicles)	Stilo (18 vehicles)
Strada (18 vehicles)	Tempra (18 vehicles)	Tipo (18 vehicles)
Ulysse (18 vehicles)	Uno (18 vehicles)	Uno 45 (18 vehicles)

sort by: Year | Price | Mileage | Colour
1-18 of 18 Results

No Photo Available
Fiat Brava (2007)
Mileage: 10000 Bodytype: Estate Colour: Silver Location: Cavan
this is a very nice car indeed its fast cheap and you will get many chicks driving it so you should seriously conside...
€974,700
More Info

No Photo Available
Fiat Brava (2006)
Mileage: 15000 Bodytype: Coupe Colour: Orange Location: Leitrim
this is a very nice car indeed its fast cheap and you will get many chicks driving it so you should seriously conside...
€949,200
More Info

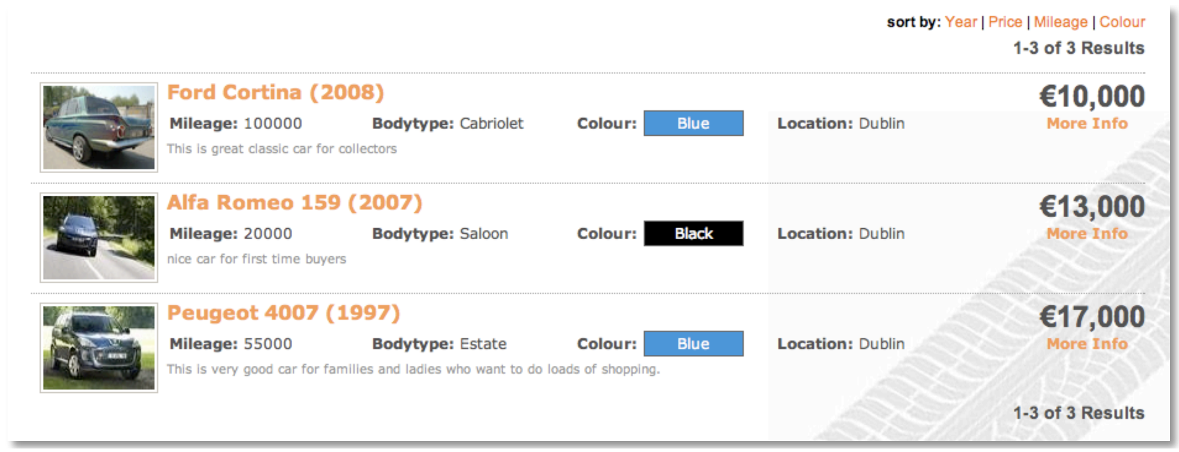
Buyer chooses car maker - FIAT
Buyer chooses car model - Brava
Search results are returned for chosen make & model

(Image 13) example of browsing for car

4.2.4 Search Results

The main issue with existing sites' search results is that they are cluttered. This problem makes results unreadable or does not present right and most important data first. By using partials (read on chapter 5) MoMo guarantees clean and very self-explanatory view of returned cars. *Image 14* shows three cars returned for some query.

Each car data is presented as high row with the most important data only, everything else is presented after clicking on vehicle's name. Notice how price is very big, car Make & Model contains year in it and image comes first.
Secondary data are: Millage, Body type, Colour and Location with headings in bold text for drawing attention. Lease important data - car short description is presented with smaller and lighter font.


sort by: Year | Price | Mileage | Colour
1-3 of 3 Results

Ford Cortina (2008)
Mileage: 100000 **Bodytype:** Cabriolet **Colour:** Blue **Location:** Dublin
This is great classic car for collectors
€10,000
More Info

Alfa Romeo 159 (2007)
Mileage: 20000 **Bodytype:** Saloon **Colour:** Black **Location:** Dublin
nice car for first time buyers
€13,000
More Info

Peugeot 4007 (1997)
Mileage: 55000 **Bodytype:** Estate **Colour:** Blue **Location:** Dublin
This is very good car for families and ladies who want to do loads of shopping.
€17,000
More Info

1-3 of 3 Results

(Image 14) example of search results

Currently three results fit on one page so system did not paginate results.
In case of multiple cars being returned pagination shows current page number plus available pages on the left top corner of returned results.
Sort options are available in top right corner for users convenience, when sort option is chosen arrow will indicate that.

4.3. The Tagging

The tagging allows easily group vehicles together by different criteria and users' preferences. It redefines the way people can search for vehicles.

4.3.1 Tag search

The way MoMo does keyword assignment is completely new idea in the car websites. It brings two new ways of finding cars. Whole concept and implementation of tagging is described in chapter 5 of this document. Here the author focuses on showing off how people could 'play' with the system or find exactly what they want through they very own queries. In fact earlier mentioned Head menu's text field lets people type in makes and models, what people are typing in are tags.

Obviously no man could possibly guess all of the possibilities, this is why MoMo comes back with autosuggestions containing the most accurate guesses for currently typed character or text. In next sections of this chapter reader will learn how different autos can be found just by using tags. Tagging is not related to earlier mentioned search (*section 4.2 of this chapter*).
It is unique and independent, still outcome might be the same but tags are giving buyers more freedom and overall experience is better.

4.3.2 Explore Feature

Tagging is available across whole MoMo, however on 'Explore' tab of the system people can view different tags grouped in categories or even join multiple keywords to achieve more accurate hits. Tags are organized in few groups called tag-categories. Table 1 shows all MoMo's tag categories.

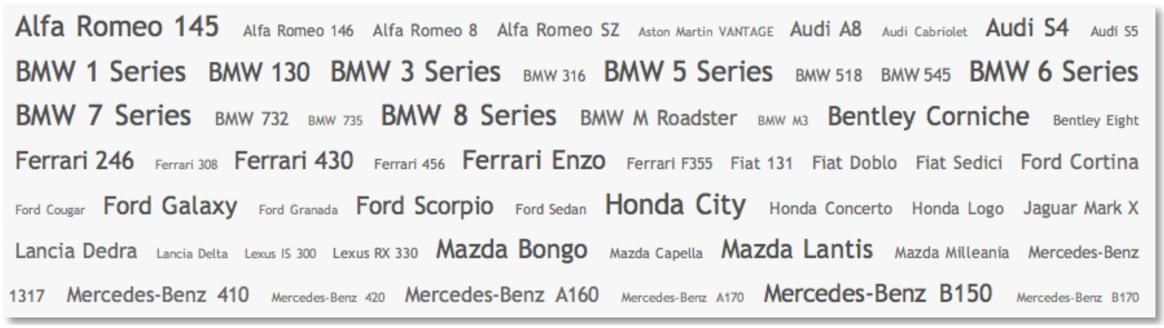
Tags Category	Description
Most Popular	The most popular tags in the system.
By Make	Displays tags corresponding to car makers
By Model	Similar like by Make but displays cars by specific models
By Price	Displays price ranges by 500 in each supported currency (Currently only Euro)
By Year	Displays available years for all cars across the system
By Location	Shows locations where cars sellers are based
By Extra	All tags corresponding to special gear like CD Player, central lock, electric mirrors, alarm, etc.
By Buyers Preferences	MoMo Offers two different ways of tagging (more in chapter 5) other category has all those tags which are selected by users. Its contain elements like: "Family Cars", "Sport Cars", "Classic Cars" etc. Which groups vehicles by they purpose or target audience/buyers and their possible preferences. (Table 1) Tag Categories

(Table 1) tag categories

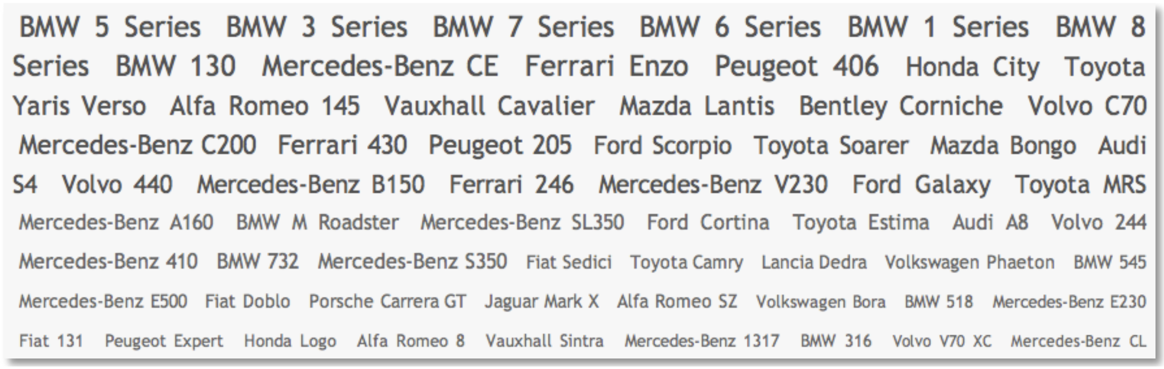
For each of categories 'Explore' tab has a link displaying a Tag cloud[2] with first few tags of particular type.
Tag Clouds are a visual way of presenting content of the site and its relevance.
MoMo generates Tag Clouds based on different criteria and allows sorting them

[2] Tag clouds are visualizations of term frequencies. A tag cloud allows you to see common terms in a text by grouping like terms together and emphasizing frequent terms.

alphabetically or by tag relevance. They come in different shapes and styles.
See *images 15* and *16*.


Alfa Romeo 145 Alfa Romeo 146 Alfa Romeo 8 Alfa Romeo SZ Aston Martin VANTAGE Audi A8 Audi Cabriolet Audi S4 Audi S5 BMW 1 Series BMW 130 BMW 3 Series BMW 316 BMW 5 Series BMW 518 BMW 545 BMW 6 Series BMW 7 Series BMW 732 BMW 735 BMW 8 Series BMW M Roadster BMW M3 Bentley Corniche Bentley Eight Ferrari 246 Ferrari 308 Ferrari 430 Ferrari 456 Ferrari Enzo Ferrari F355 Fiat 131 Fiat Doblo Fiat Sedici Ford Cortina Ford Cougar Ford Galaxy Ford Granada Ford Scorpio Ford Sedan Honda City Honda Concerto Honda Logo Jaguar Mark X Lancia Dedra Lancia Delta Lexus IS 300 Lexus RX 330 Mazda Bongo Mazda Capella Mazda Lantis Mazda Milleania Mercedes-Benz 1317 Mercedes-Benz 410 Mercedes-Benz 420 Mercedes-Benz A160 Mercedes-Benz A170 Mercedes-Benz B150 Mercedes-Benz B170

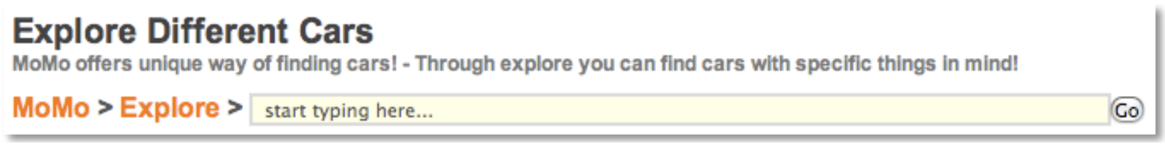
(Image 15) Tag Cloud – sorted alphabetically


BMW 5 Series BMW 3 Series BMW 7 Series BMW 6 Series BMW 1 Series BMW 8 Series BMW 130 Mercedes-Benz CE Ferrari Enzo Peugeot 406 Honda City Toyota Yaris Verso Alfa Romeo 145 Vauxhall Cavalier Mazda Lantis Bentley Corniche Volvo C70 Mercedes-Benz C200 Ferrari 430 Peugeot 205 Ford Scorpio Toyota Soarer Mazda Bongo Audi S4 Volvo 440 Mercedes-Benz B150 Ferrari 246 Mercedes-Benz V230 Ford Galaxy Toyota MRS Mercedes-Benz A160 BMW M Roadster Mercedes-Benz SL350 Ford Cortina Toyota Estima Audi A8 Volvo 244 Mercedes-Benz 410 BMW 732 Mercedes-Benz S350 Fiat Sedici Toyota Camry Lancia Dedra Volkswagen Phaeton BMW 545 Mercedes-Benz E500 Fiat Doblo Porsche Carrera GT Jaguar Mark X Alfa Romeo SZ Volkswagen Bora BMW 518 Mercedes-Benz E230 Fiat 131 Peugeot Expert Honda Logo Alfa Romeo 8 Vauxhall Sintra Mercedes-Benz 1317 BMW 316 Volvo V70 XC Mercedes-Benz CL

(Image 16) Tag Cloud – sorted by keyword relevance

Both tag clouds presented on *images 15 & 16* are consisted of links, clicking on any of them will render search results. User may preview 8 clouds in MoMo one for each tags category available. (*More information about how tag clouds are rendered is in chapter 5*)

Except available clouds users are presented with text field on breadcrumb of 'explore' (*Image 17*) where multiple tags with autosuggestion returning from system are taking place.
Again typing in it will cause system to come back with some suggestions, on top of that buyer could type comma and continue adding more tags.
This is M-Tag search feature of MoMo.

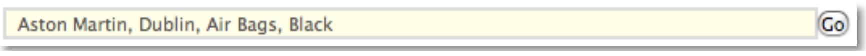

Explore Different Cars

MoMo offers unique way of finding cars! - Through explore you can find cars with specific things in mind!

MoMo > Explore > start typing here... Go

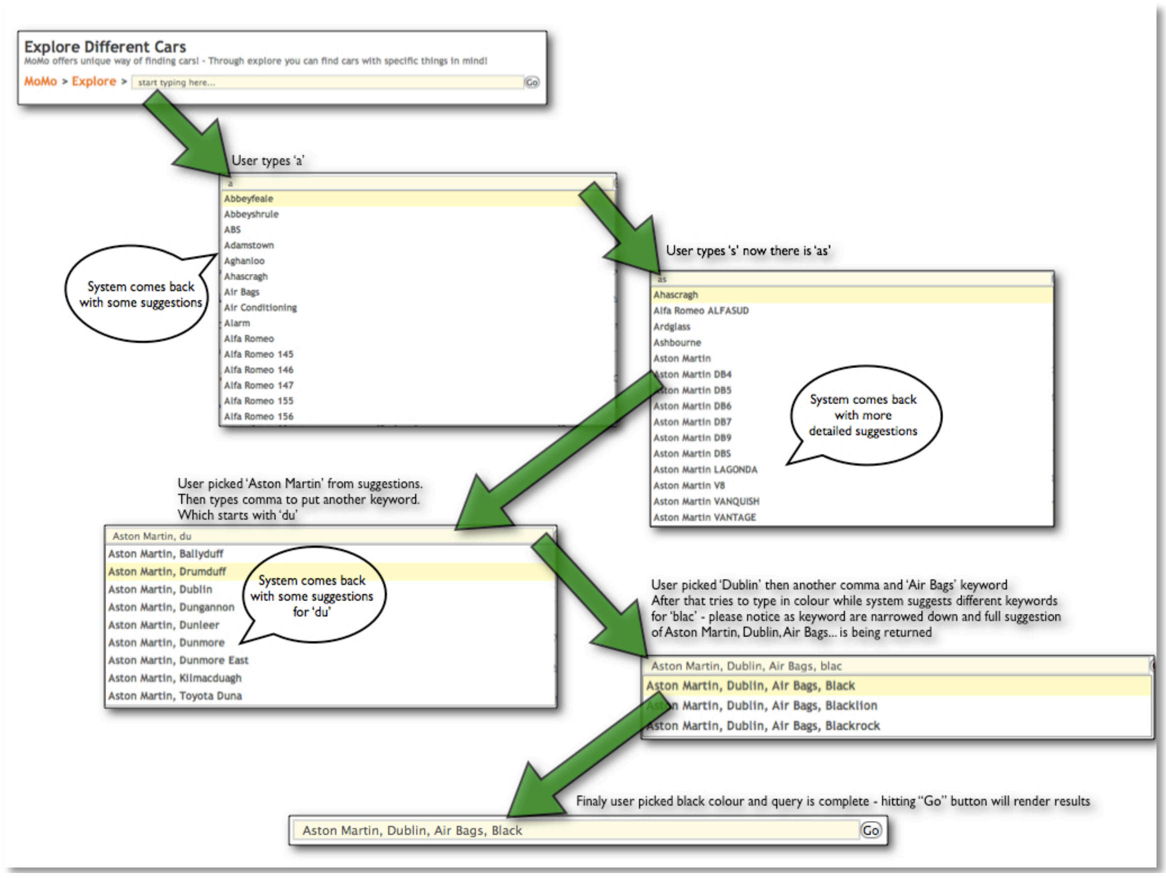
(Image 17) explore breadcrumb

Example of M-tag search where Aston Martins from Dublin, containing Air Bags and Black colour are returned is showed on *Image 18*.


Aston Martin, Dublin, Air Bags, Black Go

(Image 18) M-Tag, multiple tags choosen

All steps to type in such a query are demonstrated on *Image 19*.


Explore Different Cars
MoMo offers unique way of finding cars! - Through explore you can find cars with specific things in mind!
MoMo > Explore > start typing here... Go
User types 'a'
System comes back with some suggestions
User types 's' now there is 'as'
System comes back with more detailed suggestions
User picked 'Aston Martin' from suggestions. Then types comma to put another keyword. Which starts with 'du'
System comes back with some suggestions for 'du'
User picked 'Dublin' then another comma and 'Air Bags' keyword After that tries to type in colour while system suggests different keywords for 'blac' - please notice as keyword are narrowed down and full suggestion of Aston Martin, Dublin, Air Bags... is being returned
Finaly user picked black colour and query is complete - hitting "Go" button will render results
Aston Martin, Dublin, Air Bags, Black Go

```
a
Abbeyfeale
Abbeyshrule
ABS
Adamstown
Aghanloo
Ahascragh
Air Bags
Air Conditioning
Alarm
Alfa Romeo
Alfa Romeo 145
Alfa Romeo 146
Alfa Romeo 147
Alfa Romeo 155
Alfa Romeo 156
```

```
as
Ahascragh
Alfa Romeo ALFASUD
Ardglass
Ashbourne
Aston Martin
Aston Martin DB4
Aston Martin DB5
Aston Martin DB6
Aston Martin DB7
Aston Martin DB9
Aston Martin DBS
Aston Martin LAGONDA
Aston Martin V8
Aston Martin VANQUISH
Aston Martin VANTAGE
```

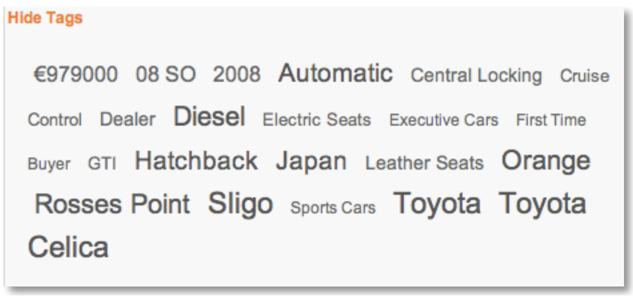
```
Aston Martin, du
Aston Martin, Ballyduff
Aston Martin, Drumduff
Aston Martin, Dublin
Aston Martin, Dungannon
Aston Martin, Dunleer
Aston Martin, Dunmore
Aston Martin, Dunmore East
Aston Martin, Kilmacduagh
Aston Martin, Toyota Duna
```

```
Aston Martin, Dublin, Air Bags, blac
Aston Martin, Dublin, Air Bags, Black
Aston Martin, Dublin, Air Bags, Blacklion
Aston Martin, Dublin, Air Bags, Blackrock
```

(Image 19) M-tag search example with demo of query building

4.3.3 Finding similar vehicles through car's Ad.

When desired car ad is finally displayed after few searches, either browsing or explore buyer may want to see other vehicles similar to current one.
MoMo's vehicles have small tags each containing keywords that are attached to them, through this simple solution we could choose any of them to see cars close to the displayed one. Please refer to *image 20* to see example of tiny cloud on car profile.


Hide Tags
€979000 08 SO 2008 Automatic Central Locking Cruise Control Dealer Diesel Electric Seats Executive Cars First Time Buyer GTI Hatchback Japan Leather Seats Orange Rosses Point Sligo Sports Cars Toyota Toyota Celica

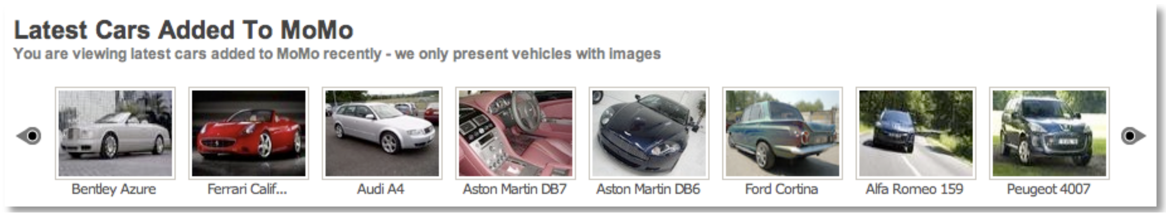
(Image 20) Vehicle ad's tag cloud for

4.4. RSS Feeds

Rather than asking people for their email addresses to send them all possible updates to their defined searches MoMo offers RSS feeds. Anyone can just register to desired type of feed and have all the latest changes showed up in their newsreader of choice.
Idea behind rss is very simple, specially formatted XML file is being rendered and proper software called reader parses it and displays data.
Usually it is just some general information without too many details and with links leading to more information about subject. MoMo offers two kinds of feeds.
They are all easily accessible by either designated RSS page or across the system, depending on their purpose.

4.4.1 RSS Feature Based Feeds

These kinds of feeds are based on certain events on the car for example, latest cars (The ones added in last 7-10 days). Example of feed being used on MoMo directly is latest cars on home page. *Image 21* shows it in more details.

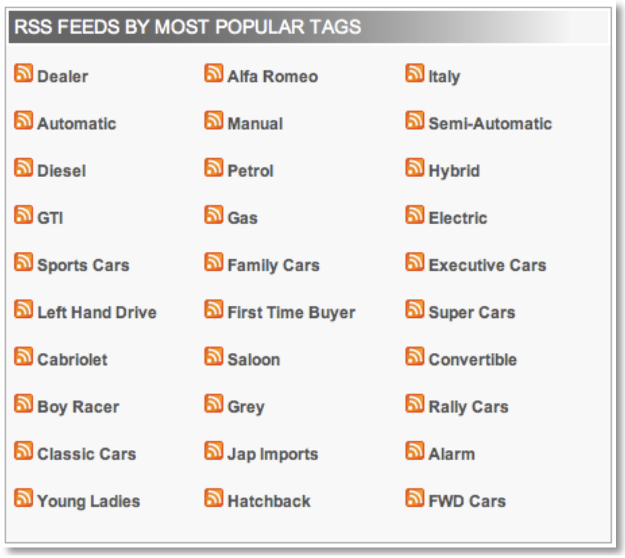

Latest Cars Added To MoMo
You are viewing latest cars added to MoMo recently - we only present vehicles with images
Bentley Azure
Ferrari Calif...
Audi A4
Aston Martin DB7
Aston Martin DB6
Ford Cortina
Alfa Romeo 159
Peugeot 4007

(Image 21) Latest cars from feed shown on the first page.

That feed was rendered as HTML directly on a page and changes accordingly as XML changes every time. However it is possible for anyone to just click and get latest cars details from very same feed MoMo displays them.

4.4.2 RSS Feeds by tags

Feeds are offered by tag name on completely separate page, they come sorted with the most popular tags first, user may as well type in their own tag or set of tags they want to observe. Same principal as during explore feature and Head menu quick make & model search applies. To reach Feeds by tag page follow link 'RSS Feeds' on Head menu (*Head Menu is mentioned in navigation section 4.1.1*)
Image 22 shows few popular tags coming back as set of links, user is encouraged to click them to see XML or to invoke default news reader in the operating system.
M-Tag search mentioned in 4.2.1 is available for user on RSS page too. People can define any query and system will render feed for such a query. So buyers are given choice of observing any change on car sales. Example All Alfa Romeos in Dublin, which are red, and from 2007.


RSS FEEDS BY MOST POPULAR TAGS

Dealer	Alfa Romeo	Italy
Automatic	Manual	Semi-Automatic
Diesel	Petrol	Hybrid
GTI	Gas	Electric
Sports Cars	Family Cars	Executive Cars
Left Hand Drive	First Time Buyer	Super Cars
Cabriolet	Saloon	Convertible
Boy Racer	Grey	Rally Cars
Classic Cars	Jap Imports	Alarm
Young Ladies	Hatchback	FWD Cars

(Image 22) RSS Feeds by top tags

4.5. Garage as sellers interface

MoMo delivers so called garage to let sellers easily maintain their stock. Garage's functionality is organized in different tabs. Read on to see differences between dealers and single time sellers garage features set. Main reason behind dividing garage in 'two' types was to let dealers have little bit extra and no limits on amount of cars they can sale.
Private sellers are offered with only three simultaneous sales at the time.
Both groups of sellers have:

- Cars in Stock – shows all cars for sale (they might be off the market)
- Sold Cars – shows all cars which are sold
- Summary – contains summary for both sold and stock vehicles.
- My account – for maintaining all seller account's details

On top of that each group of sellers has little differences mentioned later in this section.

4.5.1 Cars Sales concept, difference between private sellers and dealers

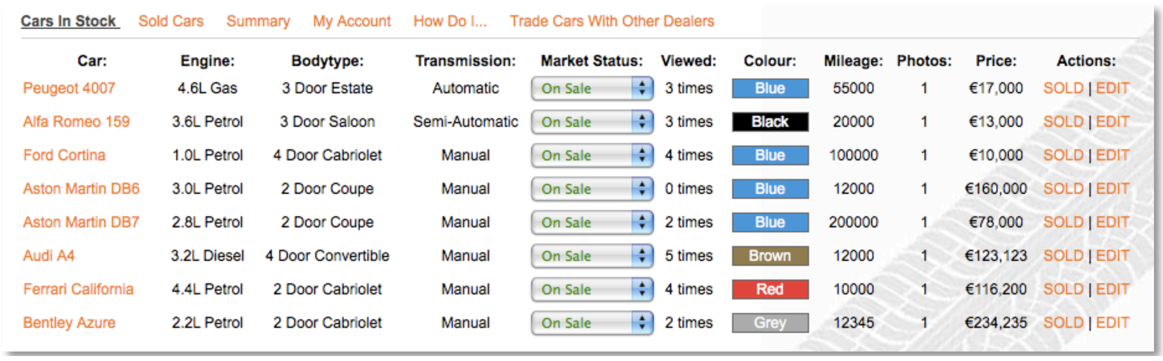
Dealer in MoMo is presented with some additional links on his menu,

- How Do I… - Leads to screen casts where dealers can learn about every single aspect of the system through nice short movies.
- Trade Cars With Other Dealers – As name suggests this feature allows dealers to trade cars between themselves. According to chats and suggestions with dealers in Ireland it is suppose to be very anticipated feature.

Menu is not only difference. When car is being added it is placed on market for certain time – this ensures MoMo does not try to sell cars, which are already sold.
Example someone comes in to the system registers and puts cars on sale, after a week or two cars is sold by some other means and this user does not bother to comeback and mark it as sold, people who will try to buy it will be confused.
For this reason MoMo has some default times on market. It is 14 days for single time seller after which he can renew his/her ad for another week.

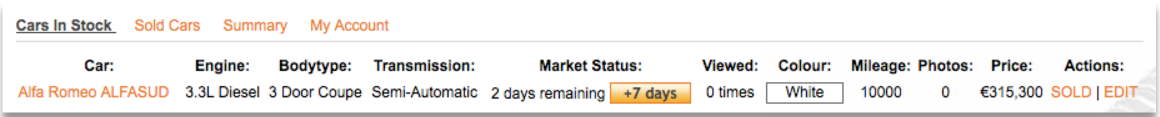
A dealer on the other hand is supposed to maintain his stock constantly.

Author believed there was no reason to put such a limit on dealers.
Still they may take cars on/off the market by simply switching drop down value. [6]
Please refer to images 23 and 24 to compare two garages car's row and on_market concept.


Cars In Stock | Sold Cars | Summary | My Account | How Do I... | Trade Cars With Other Dealers

Car:	Engine:	Bodytype:	Transmission:	Market Status:	Viewed:	Colour:	Mileage:	Photos:	Price:	Actions:
Peugeot 4007	4.6L Gas	3 Door Estate	Automatic	On Sale	3 times	Blue	55000	1	€17,000	SOLD | EDIT
Alfa Romeo 159	3.6L Petrol	3 Door Saloon	Semi-Automatic	On Sale	3 times	Black	20000	1	€13,000	SOLD | EDIT
Ford Cortina	1.0L Petrol	4 Door Cabriolet	Manual	On Sale	4 times	Blue	100000	1	€10,000	SOLD | EDIT
Aston Martin DB6	3.0L Petrol	2 Door Coupe	Manual	On Sale	0 times	Blue	12000	1	€160,000	SOLD | EDIT
Aston Martin DB7	2.8L Petrol	2 Door Coupe	Manual	On Sale	2 times	Blue	200000	1	€78,000	SOLD | EDIT
Audi A4	3.2L Diesel	4 Door Convertible	Manual	On Sale	5 times	Brown	12000	1	€123,123	SOLD | EDIT
Ferrari California	4.4L Petrol	2 Door Cabriolet	Manual	On Sale	4 times	Red	10000	1	€116,200	SOLD | EDIT
Bentley Azure	2.2L Petrol	2 Door Cabriolet	Manual	On Sale	2 times	Grey	12345	1	€234,235	SOLD | EDIT

(Image 23) Dealers garage with cars on sale with 'Market Status' column

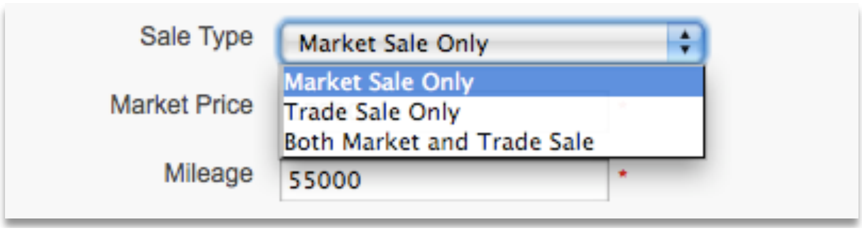

Cars In Stock | Sold Cars | Summary | My Account

Car:	Engine:	Bodytype:	Transmission:	Market Status:	Viewed:	Colour:	Mileage:	Photos:	Price:	Actions:
Alfa Romeo ALFASUD	3.3L Diesel	3 Door Coupe	Semi-Automatic	2 days remaining +7 days	0 times	White	10000	0	€315,300	SOLD | EDIT

(Image 24) Single time seller garage with cars on sale with 'Market Status' column only allowing refreshing when below 3 days on market sale. Orange button lets refresh for another week.

4.5.2 Trade Mode - dealers only market

Every dealer in MoMo may decide between selling his cars to general public, other dealers or both. It is called sale type in the system and will be referred by that name later on.
When dealer and only dealer adds/edits a car there is a drop down menu allowing him to choose type of sale for a current vehicle. (See image 25)
It is set to 'Market Sale Only' and invisible for single time sellers.
No matter what they can only present their stock to general public no other dealers.


Sale Type	Market Sale Only
Market Price	
Mileage	55000 *

(Image 25) Sales type on car add/edit

Following sales type choice 'Market price' field may be taken away and new field called 'Trade Price' may occur on the form. Therefore there are three types of sales for dealers:

- Market Sales Only – Vehicle is available to general public only and Market Price is what car's cost is.

- Trade Sale Only – This vehicle will be visible only by other dealers and no private person can buy it. 'Trade price' field will occur allowing dealer to specify price.

- Both, Market Sale and Tradable – Vehicle can be bought by both private buyers and dealers across the system, 'trade price' and 'market price' are available allowing to set different price for each type of sale.

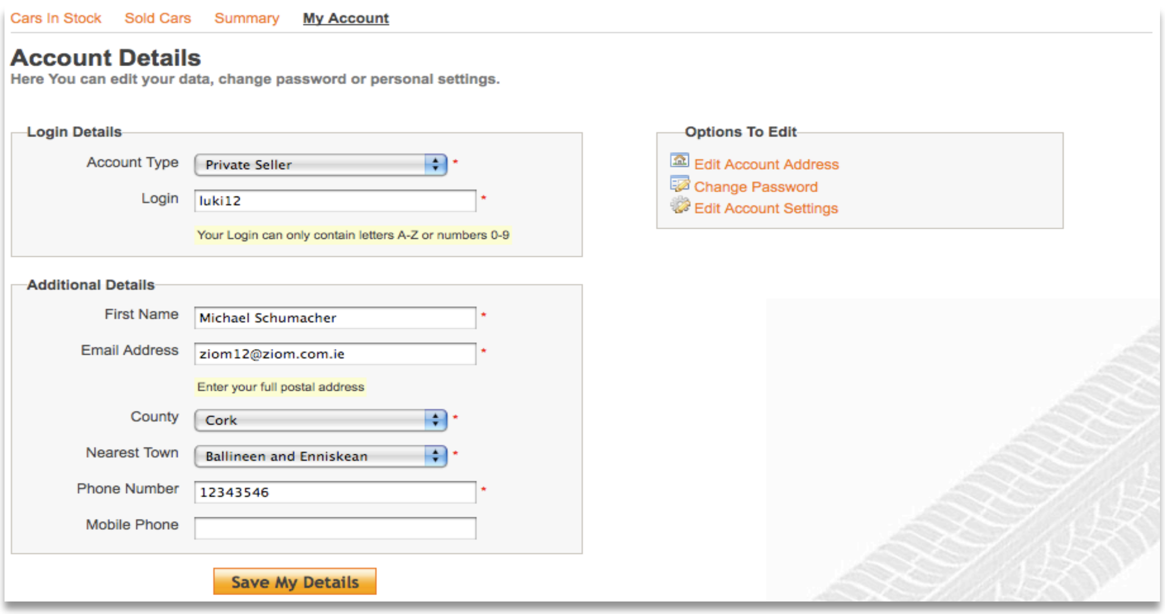
Each dealer may then perform searches for other dealers' cars available for trade.
This is only accessible from dealers garage last tab so no normal buyer or private seller could trade cars or find tradable cars. As author gathered feedback it turned out that lots of dealers would be very happy with such a feature.
According to them it reassures they can restore stock quickly by buying cars from other dealers.

4.5.3 Account management

In account management tab user can:

- Change password
- Modify Account details like address or contact information
- Edit account setting – allowing mails from interested in buying cars people etc.

Consult image 26 to see how account details look like for private seller.
For dealers they look nearly the same except some additional data is required.
Dealers need to put in Company name and address of their business too.



Account Details
Here You can edit your data, change password or personal settings.

Login Details

Account Type	Private Seller *
Login	luki12 *
	Your Login can only contain letters A-Z or numbers 0-9

Additional Details

First Name	Michael Schumacher *
Email Address	ziom12@ziom.com.ie *
	Enter your full postal address
County	Cork *
Nearest Town	Ballineen and Enniskean *
Phone Number	12343546 *
Mobile Phone	

(Image 26) My Account tab for private seller

5. Implementation of MoMo's features

Chapter describes in detail implementation of main functionality in MoMo.
Its design, searches implementation including detailed coved of different tagging methods and their usage. Code, libraries and tools with frameworks for user interface enhancements.
Innovative ways the MoMo uses for configuring different aspects of the system are covered in "Configuration of MoMo" subsection and followed by search engine optimization techniques.
Next m-tag search is disassembled to smaller parts and explained carefully.
Chapter finishes with presentation of different types of caching and their maintenance including sweeping.

5.1. MoMo's design

Design of application started from database and class definitions, this schema is available in Appendix C. (Data base schema in MoMo). Author defined all references between different models and identified main actions each of them will be performing or supporting.
MoMo fully follows all modern principals of applications design like:

- Code reusability – by trying to incorporate all common help methods in application helper all views across whole MoMo can use it anytime at minimal cost and without repeating code in multiple places. Additionally partials let reuse certain HTML, JS, CSS elements to be written once and reused as other view sub elements or on their own.

- DRY – Principle whole rails is based on. DRY stands for "Don't Repeat Yourself "

- MVC oriented – RoR enforces certain shape of web application based on it.

- Every single file type and class belongs to Model, View or Controller on top of that some of them belongs to supportive types as helpers or migrations. This guarantees consistency and cleanness of produced code base.

- Different plugins and add-ons are completely independent.

- User's interface simplicity and consistency across whole app

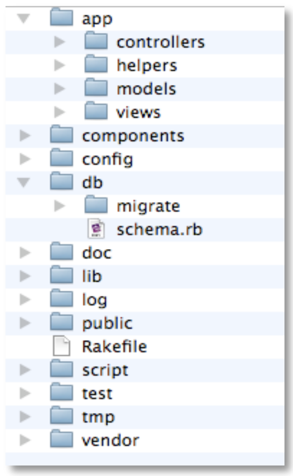
- Restful – all links are actually meaningful and clean without ugly parameters.
  Ensures clean URIs for end users and bookmarking services.
  Enhances security, as it's harder to perform any kind of injection because parameters passed to controllers are not exposed.

5.1.1 Directory structure

Directory structure in this auto portal is organized as standard Rails application. Image 27 shows that clearly.

app – contains subdirectories for controllers, helpers, models and views

components – some additional elements of application's are stored here, however MoMo does not use it.



```
app
  controllers
  helpers
  models
  views
components
config
db
  migrate
  schema.rb
doc
lib
log
public
Rakefile
script
test
tmp
vendor
```

(Image 27) Directory structure

config – directory stores all configuration elements for application. Different setups and environments for running rails application included.

db – everything database related lives here, schema.rb generated off the migrations and migrations themselves in migrate subdirectory.

doc – documentation auto generated based on rdoc tags in code (not used)

lib & log – additional libraries shared across whole application and logs from app servers

public - static content like CSS, JS and images

script – scripts for running application, server, console and some other command line tools.

test – directory has different unit tests from application

tmp – temporary files directory, in MoMo used for storing temporary uploads.

vendor – all third party plugins and add-ons land here.

5.1.2 File types and their purpose

Rails framework is fully MVC stack based, main elements are:

- Models – ORM (Object relational mapping) for connecting data in db with application logic.

- Controllers – Ensures communications between different elements of the system. Gathers data from views for different actions in backend or talks to models and passes processed data back to views for rendering response.

- Views – written in different template languages are what browse gets back after controller delivers information back. Read on different file types for understating different types of views, reasons and purpose in Memo.

Ruby on Rails is mature framework and in addition to Models, Views and Controllers from MVC pattern it has few other files types with different roles in the system.
Some of them come default bundled with rails while others were added as plugins or gems.

- Migrations – files, which allows developers to create their schema, details in ruby.

Contain not only details on data types hold in DB but during first time DB setup injecting some constant data into database server. (Towns, Makes, Models, etc.)
- Helpers – contain variety of different methods used across multiple views – ex. Dropdowns for counties or towns come in edit/create account.

- SASS files – installed as third party plugin language for generating CSS supporting variables definition and rendering optimal CSS.

- Partials – partials are actual view elements of MVC however they can be rendered in any other view or on their own just by controllers method, as a result they decrease amount of code in project and increase reusability.
- Routing – routing in rails is configured through file *routes.rb*. It supports multiple rewrites and action mappings. Lets developers define custom parameters and their default values when nothing is being passed.

- Views – View element of MVC in MoMo is made of different templates:
  - XML Feed responses – for RSS requests
  - RJS responses for dynamic DOM manipulation – progress indicators or elements removal.
  - HAML – XHTML rendering language where as in Ruby language indents are used instead of brackets for code blocks.
  - RHTML – standard build in rails library for building up interface.
  - Inline html – when controller renders simple HTML snippet on their own. Later its injected somewhere in existing page – difference between inline html and RJS is that RJS tries to manipulate by using JS prototype library where inline is just like text response with designated holder in dom.

5.1.3 Rake tasks for maintenance and setup

As mentioned in tools section rake is simply make like program in rails.
The framework comes with some default tasks bundled with it, which can be easily performed by calling them as a parameter to rake command. Different rake commands are grouped together in calls:

- rake db:migrate - migrates all migrations (read on next section) to DB
- rake db:drop - drops database content
- rake db:create - creates database based on schema file

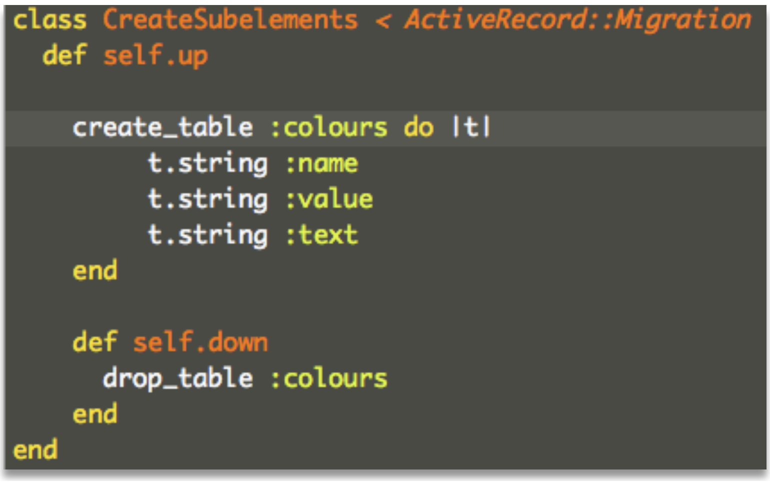
Additionally MoMo's author comes up with his own tasks for specific business logic:

- rake momo:expire_cars - check cars off market date and expires cars that passes that date
- rake momo:test - runs some MoMo specific tests on the application

Rake tasks are nicely organized by type. Above examples are either db for database or momo for MoMo's specific logic. Then they have name explaining tasks aim.
It is not uncommon to have more than one semicolon for types and subtypes like:
rake db:sessions:clear – cleans sessions table in db for dropping all old sessions.

5.1.4 DB schema and migrations

MoMo uses migrations to get DB schema exported to MySQL server. Migrations are part of rails framework and let easily define data types and all their limits directly in ruby language.
All migrations together are rendering *schema.rb, which* is exact equivalent off all migrations together. The schema could be later reused through RAKE tasks to recreate DB structure.



```
class CreateSubelements < ActiveRecord::Migration
  def self.up

    create_table :colours do |t|
         t.string :name
         t.string :value
         t.string :text
    end

    def self.down
      drop_table :colours
    end
end
```

(Image 28) Migration for creating colours

As shown on *image 28* migration is ruby defined table definition, mentioned example shows colours table being created on migration and dropped on 'down' rake call.
Colours table has three columns: name, value and text that are all string type corresponding to varchar(100) in database. Primary key is auto created and there is no need of specifying it here.
This is just simple example, migrations are very powerful and let specify every aspect of SQL table's column.

Please refer to *Appendix C. (Data base schema in MoMo)* for detailed DB diagram.

5.1.5 Restful principles

Portal creators and websites designers in general forgot about very basic and simplistic ideas behind HTTP protocol. Current sites are overloaded with parameters passed to it in very unclean manner. The main element of HTTP assumes that browser invokes actions on the web server.
This does not mean meaningless URLs and URIs unfortunately this is what most websites offer now days. It reaches far beyond car market, every bigger website faces this issue, either eBay or hotmail.

Restfulness of MoMo means meaningful urls and proper actions invoked on the server by just passing GET, POST or not implemented in modern browser but specified in HTTP protocol standard DELETE and PUT. Thanks to those actions different URL could be reused and let achieve different aims depending on what way browser will invoke it.

Table 2 shows examples of what links usually are and how they look in MoMo, please notice that some of them are fictional and created for sake of this thesis.

Commonly used links	Links MoMo offers
http://momo.ie/explore?tag=supercars	http://momo.ie/explore/supercars
http://momo.ie/explore?tag=supercars&tag2=dublin &tag3=black	http://momo.ie/explore/supercars,dublin,black
http://momo.ie/user?action=login	http://momo.ie/login or ../session/new

(Table 2) URL Mappings

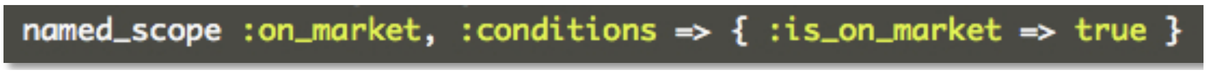
5.2. Implementation of search

MoMo has variety of methods for finding a vehicle, they are not dependant on each other and implementation varies too. Following section focuses on explaining different implementation of every system's search elements. Excluding tags that are described in detail in section 5.4 of current chapter.

5.2.1 Simple search aka quick search

For performance reasons MoMo renders javascript array of available vehicles, array contain make, model and every year for each model. This is called dynamic_mmy and allows quickly change content of model or/and year drop downs depending on make.
Second purpose is that it reassures that only cars on_market (*see garage section in chapter 4*) are allowed to be in menu. This way system avoids situation where buyers get empty search results as they search for non-existing stock. Dynnamic_mmy is being cached after JS file renders it stays in public directory for 30-60 minutes depending on dealers' activity or system load. Caching guarantees no application server involvement in dynamic_mmy delivery except when cache has to be re-rendered.
On Controllers side of things simple search is using cars_controller and performs simple active record search. The portal has special type of model. It doesn't inherit from rails' ORM ActiveRecord. Called Finder.rb is responsible for most simple searches operations.

All searches across whole system are using rails 2.0 feature called named_scopes.
Idea behind them is to define global finders with specific requirements in mind.
Example on Image 29 shows car.rb model named scope



```
named_scope :on_market, :conditions => { :is_on_market => true }
```

(Image 29) Named scope for cars on market.

Every time any find is performed it calls :on_market named scope first and another parameters are passed and filter out specific cars from whole set of cars, which are on the market.

For understanding this simple snippet please refer to DB schema in *Appendix C: Schemas and Big Images*. Is_on_market is Boolean column type in car table and it indicates if car is on the market as section about garage explains.

5.2.2 Advanced search aka power search

Advanced search is long form with every single field off the car available.
Users can pick anything they want and are not forced to choose anything! They may for example find specific car types and cars from within certain money ranges and year ranges.
This kind of functionality varies from what other car sites offer as they need to know at least make and model to search for. It is possible because MoMo implements Power Search as separate MVC functionality!
Advanced search has its own model with unique methods for finding anything in any circumstances users define for themselves. This allows storing searches in db guarantees that people reuse them and bookmarks will be made of small links.

For optimal design MoMo has multiple dropdown generators in application helper (its being used across whole application) So on car edit/add sellers actually use the very same dropdown values people will define their searches in on power search form.

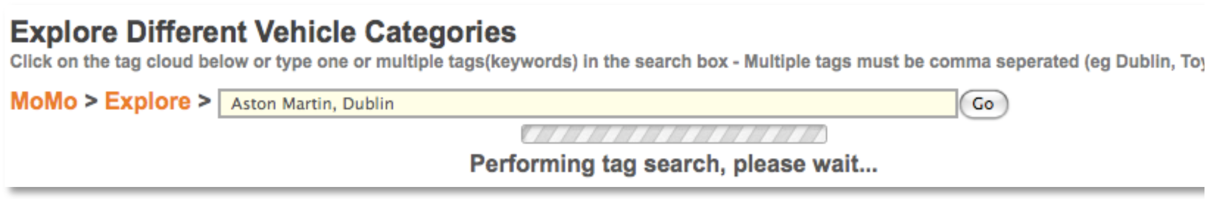
5.3. User interface enchantments

The MoMo owes most of its user-friendly interface's features to prototype library and different web patterns, which this chapter covers. Author spent some quality time testing different solutions and ideas for photos manipulation and storage in web applications.

5.3.1 Implementation of images compression, libraries

The system lets sellers upload 15 high quality photos. Each image has thumbnail generated and becomes compressed, optimized for supported by MoMo size and downgraded in quality to achieve lower sizes and higher performance in delivery yet look nice.
All this functionality is performed by ImageMagick open source library, which can do possibly any image manipulation we could think of. To make it work with rails and to do it in efficient manner, author decided to user mini-magick GEM. It is little ruby wrapper around ImageMagick and allows direct calls to ImageMagick API on OS level. It is small, very efficient and directly calls C API of ImageMagick, therefore code for supporting it is entirely placed in separate model called *Photo.rb.* During work on MoMo RMagick library was tested as well, however its big memory footprint disqualified it for production usage in portal of this magnitude.
Performance tests for accessing images proved that vehicle's photos are best stored on hard drive rather than in database blob column type.
Model's filters before_create and before_delete guarantees saving image on models image creation and deleting actual file on model removal.

5.3.2 UI progress indicators


Explore Different Vehicle Categories

Click on the tag cloud below or type one or multiple tags(keywords) in the search box - Multiple tags must be comma seperated (eg Dublin, To

MoMo > Explore > Aston Martin, Dublin Go

Performing tag search, please wait...

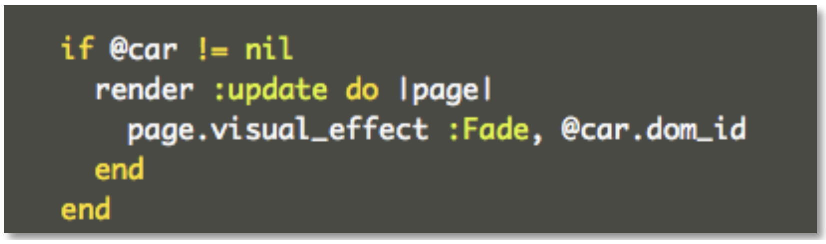
(Image 30) Progress indicator for tag search on Explore in MoMo

User interface progress indicators like one on *Image 30* are for users convenience so people know that actually system performs their search after their click search/explore/go button.
When some very complex M-Tag searches are performed they make take up to couple of seconds and experience with people testing applications shows that they become border and confused by nothing happening, this simple progress indicator reassures that query is being performed.
This feature is used in few places in MoMo including photo uploads progress indicators, mentioned above explore M-tag search and RSS feeds generator for custom tags too.
It's implementation is actually very easy and probably well known for quite a few years right now, definitely earlier than web 2.0 authors come up with their principles and ideas.
Page contains hidden div that contain image (animated gif) and text "*Performing you action, please wait*" which obviously vary depending on context. When user click button or links it has little java script function calling that div by id and switching it to become visible.
After page reloads when results are back it disappears back into hidden state on its own.

5.3.3 AJAX calls for maintaining stock in garage

In tools section of this thesis reader can learn about the prototype library being used across MoMo.
Prototype implements AJAX object with methods like Request and Update for different operations. Developer may easily call them for updating/modifying some DOM elements or just perform some backend changes when controller gets those requests.

Good example of situation where *Ajax.Request* is being used is sellers' interface – the garage.
When user sells car it moves to sold cars section instantly, what really happens is that each row of cars in garage has unique ID and when "sold request" is sent to controller and after controller makes proper changes through car model. Little RJS template comes back with modifications for DOM. User's browser resolves all the actions it received in RJS and is ready for another requests.
Important thing here is that RJS removes element only when backend performs its' tasks successfully. Big win for web developer is that Ruby shields whole Prototype calls behind ruby helper methods.



```ruby
if @car != nil
  render :update do |page|
    page.visual_effect :Fade, @car.dom_id
  end
end
```

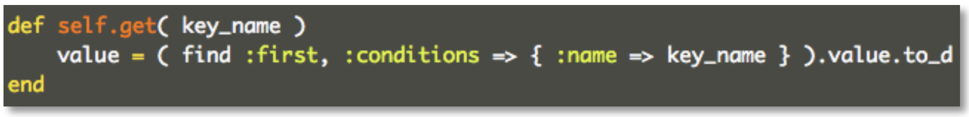
(Image 31) Ruby snippet renderer for DOM manipulation

Code from *Image 31* is showing what happens when car is marked as sold.
Page is object in rails - it renders proper action in prototype js behind the scenes and such a code is browser independent. On top of that developer may specify some simple UI effects in this case DOM element will FADE off the page which again increases user's experience.

5.4. Configuration of MoMo

MoMo uses separate Model called *Settings.rb* where additional data regarding system's behaviour is set. This model supports special getter for obtaining information and auto boxing to proper format (decimal). Each bit of the information is stored as separate row in database.
Main advantage of this is real time system's configuration – there is no need to restart servers for changes to take effect. Most data is just name and number consult examples:

- Number of rows per search results
- Car model validation details (min and max values for engines, years, doors, etc.)
- Number of allowed photos per car
- Amount of time different sellers may keep their vehicles on market without refreshing
- Number of tags per bucket



```ruby
def self.get( key_name )
    value = ( find :first, :conditions => { :name => key_name } ).value.to_d
end
```

(Image 32) getter in settings for resolving application configuration

MySQL per query caching ensures that it won't slow down application.

5.5. Tagging

All the biggest companies in the industry like: Google, Yahoo, AOL or Microsoft are using indexes to hold data about harvested websites.
This lets them quickly scan and find what users ask for in a query.
What is the downside of such a solution? They need to crawl Internet constantly to gather new data about new websites and changes in existing ones.
The Task not easy at all when we consider number of websites out there and changes taking place in their content. While using indexes on single portal might seem much easier we need to remember about lag when data becomes available and is being indexed.
As indexing requires certain processing power it may interfere with normal operations of the portal, therefore natural is to have indexing running in some periods of time.
The more often data changes the more often it has to be re-indexed or added to existing index.

However there will be always small gap between what website has and whether users can find it through the site's search engine. Tagging on the other hand does not require crawling data constantly. System designer may define different types of keywords that will be assigned to object in the system during creation, update, or any other event. Portal's administrators may even allow users to participate in tagging too. This lets people to decide on their own what kind of different keywords, groups and categories of data they want.
Besides this creates natural connection as users pick tags meaningful for them.

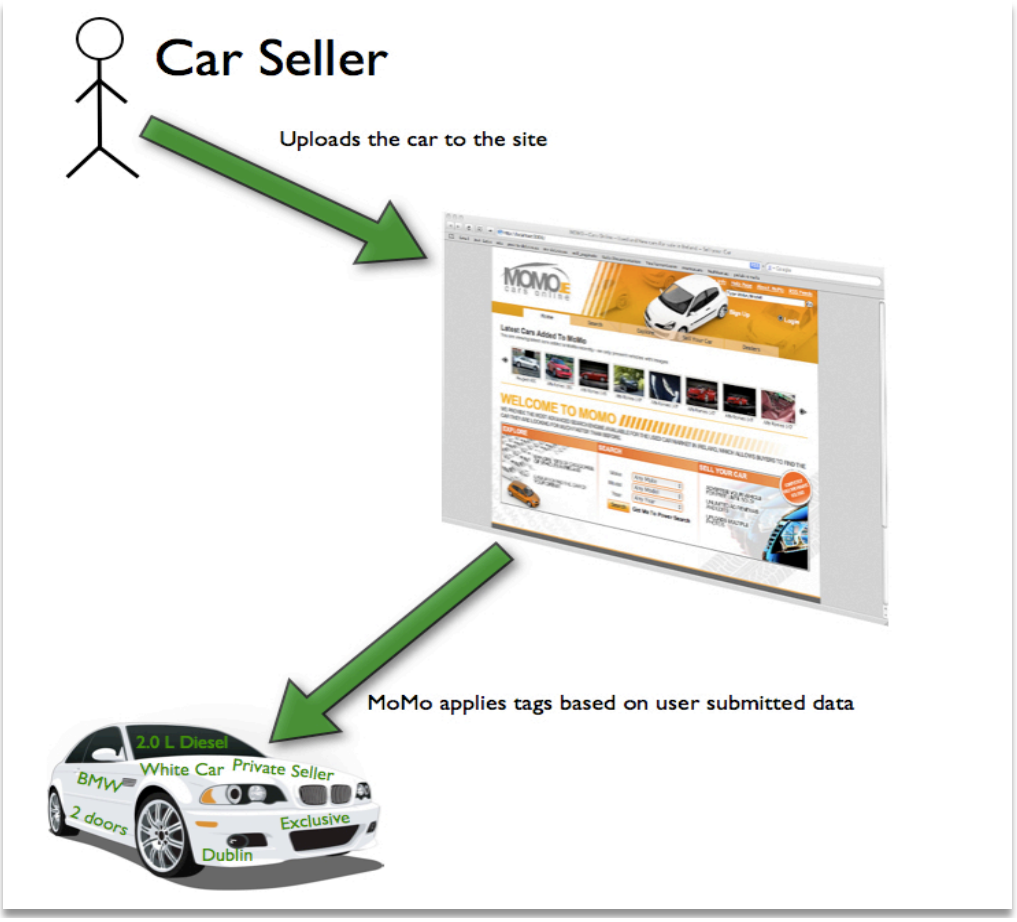
5.5.1 Folksonomy against defined tags

The folksonomy is a specific example of tagging where choice about tags is left completely to users. Good example of folksonomy usage is delicious [13] social book marking site where whole content is based on users input. Delicious aggregates different websites categorized (tagged) by people preferences and their subjective evaluation, so when searching on delicious we only find what other people found interesting and useful.
There might be no agreement between different peoples tastes and choices, but specific keywords will be very accurate to describe site's purpose or it's content.
In contrary to folksonomy system may offer defined tags and let people choose from them, its safer as people won't be given chance to abuse the system. However it limits the system in certain ways. Case of web portal oriented on car sales is quite unique, as normal visitors are not meant to have accounts or any input on the site. For that reason MoMo uses auto-tags, which are generated, based on some data dealers or private seller puts in. (Read more details about them in next section 5.3.2) Additionally sellers may pick up to three predefined so called user-tags they are described in more details later. For all those reasons portal does not support free flow tags.

5.5.2 Auto-tags vs User-tags and their implementation in MoMo

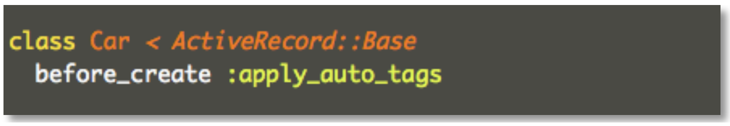
MoMo has two methods of assigning keywords to vehicles imported to the system. First one is called Auto-Tagging, as name suggests every car has some properties that are being scanned and tagged based on their values. For example:

Aston Martin DB5 from year 2005, Black Metallic from County Dublin and Adamstown village will be tagged with: Make - Aston Martin, Model - DB5, Color - Black, Location - both Dublin and Adamstown. For all of those keywords counter will be increased so exact weight is being calculated every time car is added or removed from the market.
(To see that process consult image 33a.)


Car Seller
Uploads the car to the site
MoMo applies tags based on user submitted data
2.0 L Diesel
White Car
Private Seller
BMW
2 doors
Exclusive
Dublin

(Image 33a) – Demonstration of car auto-tagging process.

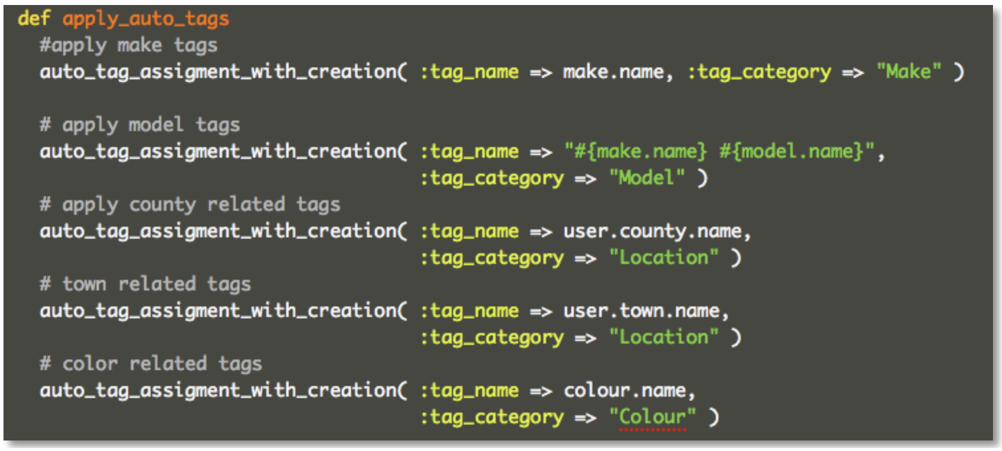
Auto tagging uses Rails ActiveRecord events observing feature.
On car model class there is an observer for particular event (*see Image 33b*) - before model's instance is created in database it executes method **apply_auto_tags**.



```
class Car < ActiveRecord::Base
  before_create :apply_auto_tags
```

(Image 33b) - code sample of ActiveRecord event observer in car's model

It is set of functions to guarantee that all cars' fields will be tagged properly.
This includes automatic recognition of foreign keys from other tables which are referred from current car and assigning tags based on their values. Consult image 34 on this page to see how **apply_auto_tags** makes multiple auto tagging calls possible by passing keyword which has to be tagged and what tag category it should fall into.

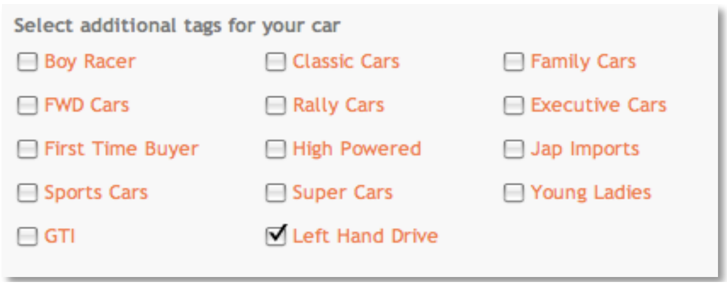


```
def apply_auto_tags
  #apply make tags
  auto_tag_assigment_with_creation( :tag_name => make.name, :tag_category => "Make" )

  # apply model tags
  auto_tag_assigment_with_creation( :tag_name => "#{make.name} #{model.name}",
                                    :tag_category => "Model" )
  # apply county related tags
  auto_tag_assigment_with_creation( :tag_name => user.county.name,
                                    :tag_category => "Location" )
  # town related tags
  auto_tag_assigment_with_creation( :tag_name => user.town.name,
                                    :tag_category => "Location" )
  # color related tags
  auto_tag_assigment_with_creation( :tag_name => colour.name,
                                    :tag_category => "Colour" )
```

(Image 34) - code sample of car's model apply_auto_tags method

Very same method is being called when car's data is updated. First check for car's market status is performed. If it has changed - all tags for that vehicle are having their market_count (*see DB schema table in Models chapter for fields description details*) value decremented accordingly. This way MoMo ensures that tag clouds are always up to date and user is being presented with current data. It is like automatic reindexing except without all the downsides.
Another way the cars can be tagged in MoMo is users' input. Vehicle seller can tag their cars when adding them to the system or through later edition. MoMo defines those tags as User Tags. They fall into "*By Buyers Preferences*" category and their main purpose is to target specific groups of buyers. At the moment MoMo has about 14 predefined user tags seller can choose from. Plans are to leave empty text field to let people add their own unique tags too.
However to prevent abusive keywords and cheating system only allows to choose from up to three of such a tags.



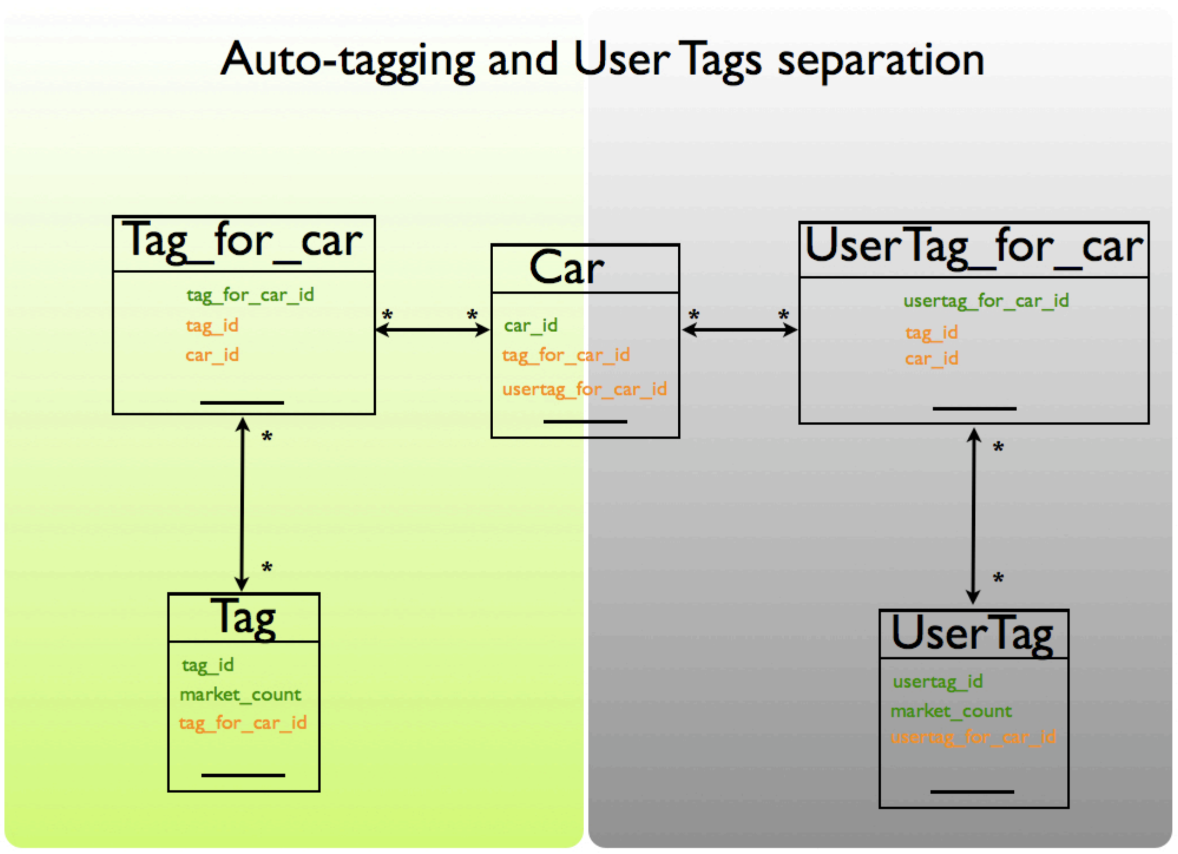
Select additional tags for your car

☐ Boy Racer	☐ Classic Cars	☐ Family Cars
☐ FWD Cars	☐ Rally Cars	☐ Executive Cars
☐ First Time Buyer	☐ High Powered	☐ Jap Imports
☐ Sports Cars	☐ Super Cars	☐ Young Ladies
☐ GTI	☑ Left Hand Drive	

(Image 35) Available Users Tags on car add/edit form

This hybrid like keyword assignment makes MoMo unique and lets car buyers to perform searches based on any criteria.
In this same time MoMo staff have a control over users tags - by making sure there won't be any illegal, unrelated and abusive content. As well it helps to react to market changes. For example new law in European Union will create tag bans, which classify cars by different levels of CO_2 emissions. Adding new option to either automatically tag cars based on that criteria or letting owners decide is a matter of adding new row of data to Database. This same applies to tag categories - ex. When Tag Bans are out adding Tag Bands category to aggregate all related tags would be perfect way to let people perform searches based on that.
In comparison to other car selling sites this is completely new quality aiming at positive experience of both buyers and sellers. Note that at the time of writing this thesis (July-August 2008) no other car system does tagging.

Both auto-tagging and usertags are completely independent from data schema point of view, to understand what it means consult *Image 35a* where different entities are showed together with the reference to the car object.



(Image 35a) Separated implementation of user-tags and auto-tags, green side shows auto-tagging while grey side to the right corresponds to user chosen tags (predefined by MoMo before hand) Green fields mean local id or property while orange mean foreign keys.

5.5.3 Tag clouds

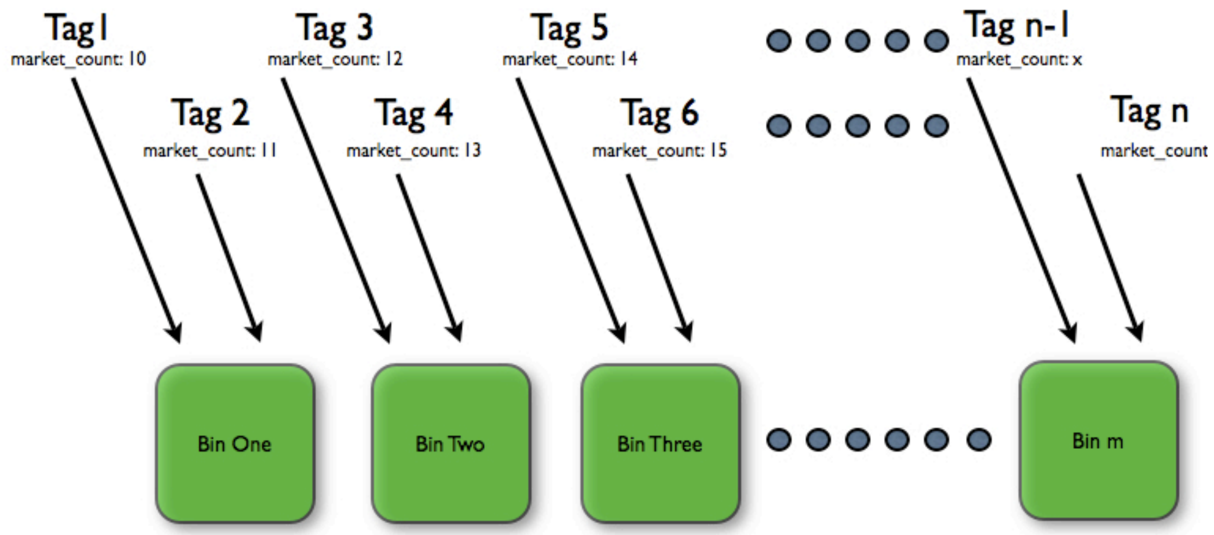
By having a certain amount of tags in the system, the portal may deliver *Tag Cloud*.
Refer to chapter 4 section *4.3.2 Explore Feature* of this document to see how tag clouds look like and behave. Here document focuses on describing ways Tag Clouds are generated, cleaned up, balanced and delivered to users.
Each tag in MoMo has different weight - which is dependent on number of cars being tagged with particular keyword. Higher weight makes keyword more important and that is the reason for some keywords to be bigger, bolder and with darker color while others are small and with light color of their font. All of them are links that will show search results containing cars corresponding to them.

The system calculates tag weights only for cars currently on sale. Tag's weight is changed depending on the vehicle's market status. The car might be on or off market. Exactly as in case of normal searches this assures that people can't find vehicles not for sale or sold.

Based on market count value for each tag and its category we can render multiple independent tag clouds. Using linear buckets algorithm system groups tags into separate containers called buckets which are then sorted internally and for each bin different CSS Style is chosen so each group of tags have different look - according to their container's significance against other buckets. In addition to this each tag cloud rendered for category is limited by top 100 most popular tags in given category.
Whole idea of tag clouds is based on assumption that first 100 tags follow a linear pattern.
Please consult plot 5.3.3 and below equation to understand how MoMo qualifies tags to buckets.

$$\text{Tags per bucket} = \frac{\text{All tags}}{\text{bucket's size}}$$

Bucket's size is set in Settings model. (*see 4.2 Configuration of MoMo subsection*)
By default portal uses bucket of size ten. For one hundred first tags there would be 10 tags per bucket. Each of those bins has CSS class for different colours, font sizes and weights.



(Image 36) Linear distribution of tags

Notice how bucket size makes two tags end up in one bin on example pictured on *image 36*.
Changing bucket sizes dynamically modifies amount of tags, the only limit is number of CSS classes to support more than default 10 buckets. This number was set as result of few experiments and tests.

5.6. Search Engine Optimization

The portal uses *content_for* directive for dynamic injecting titles and keywords in each sub-page.
As a result crawlers will have easier way to get information on each sub-page's purpose and content. In main application layout special place holder called :head and :title is in place
By using directive *content_for :head do* we could put any title, code in earlier prepared holder. Main thing for data exposure for data harvesting programs are tags.
By using microformats rel="tag" [rel. 1] inside links guarantees that crawling software will understand meaning of links as tags. See more about microformats [1]

Example of tag link in MoMo with rel for microformating:

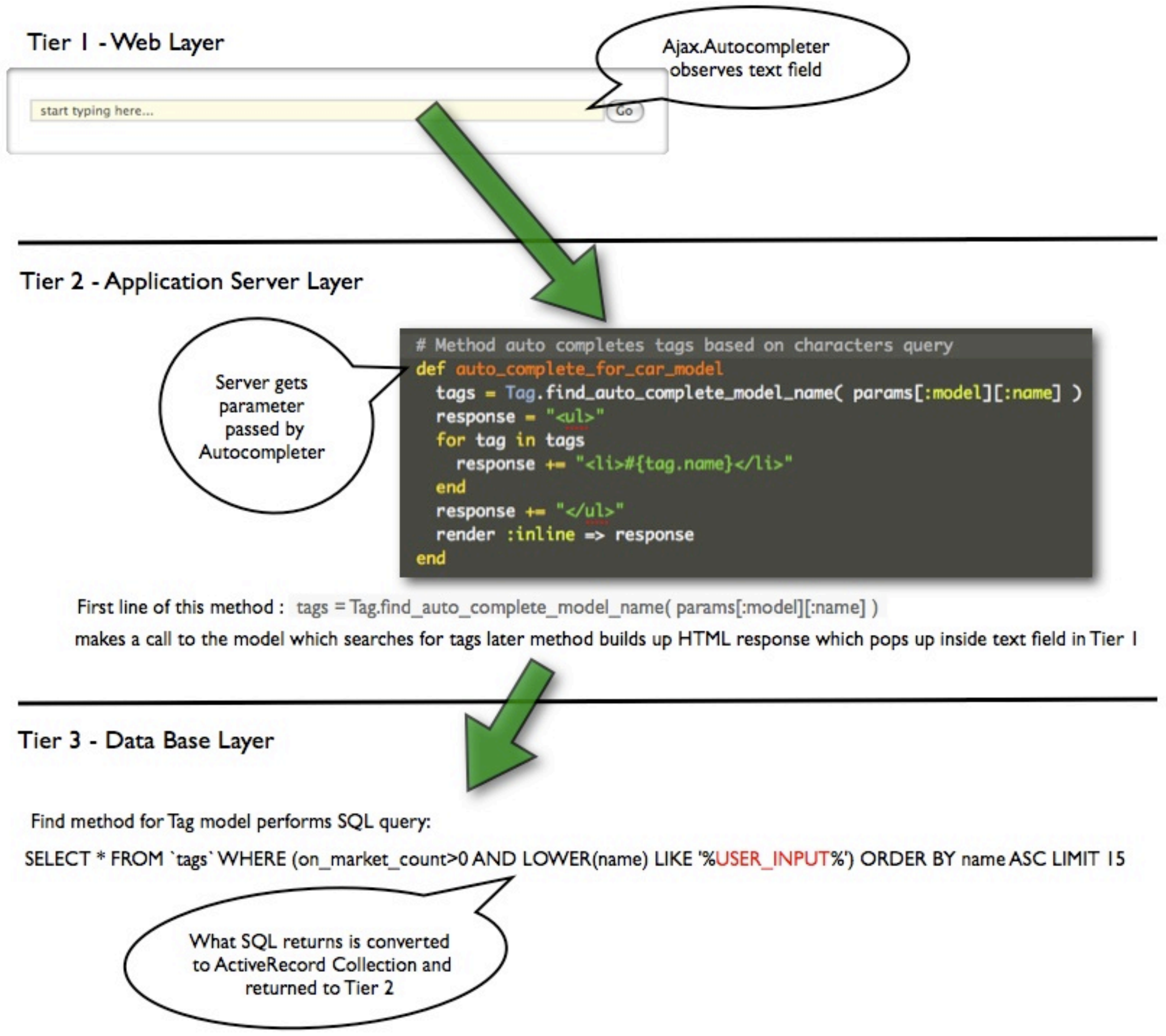
<a rel="tag" href="http://momo.ie/explore/Classic Cars" class="tag9"> Classic Cars </a>

Aiming for good site exposure in main search engines it is important to remember about few additional tricks:

- H1,H2 & H3 elements are very important for web harvesters too
- The same LI, UL & OL HTML tags
- Links from other sites to our site make our site "more visible"
- Target oriented sites will be easier to find therefore searching for "Classic Cars" people will find MoMo as it exposes this kind of market cars by special user tags mentioned in tags related section. Besides advertising might be very well targeted too.
- Web spiders gather data from tag clouds in very efficient manner.

5.7. Explore markets feature and dynamic queries with auto completion.

With all the different keywords attached to cars by MoMo and car sellers. System lets buyers to define they very own queries for whatever criteria they want.
However this kind of functionality would require people to know all the tags they can type in and search for - this for obvious reasons won't ever happen.
MoMo's author come up with an idea to auto suggest keywords as people start typing, this simple idea is based on asynchronous calls to back end every time user types a character in form's text-field.
The exact user's experience is well covered in chapter 4 section 4.3.2 *Explore Feature.*
Here author will describe how MoMo does auto completion on every layer of the application.


Tier 1 - Web Layer
start typing here...
Go
Ajax.Autocompleter observes text field
Tier 2 - Application Server Layer
Server gets parameter passed by Autocompleter
Tier 3 - Data Base Layer
What SQL returns is converted to ActiveRecord Collection and returned to Tier 2

```
# Method auto completes tags based on characters query
def auto_complete_for_car_model
  tags = Tag.find_auto_complete_model_name( params[:model][:name] )
  response = "<ul>"
  for tag in tags
    response += "<li>#{tag.name}</li>"
  end
  response += "</ul>"
  render :inline => response
end
```

First line of this method : tags = Tag.find_auto_complete_model_name(params[:model][:name])
makes a call to the model which searches for tags later method builds up HTML response which pops up inside text field in Tier 1

Find method for Tag model performs SQL query:

```
SELECT * FROM `tags` WHERE (on_market_count>0 AND LOWER(name) LIKE '%USER_INPUT%') ORDER BY name ASC LIMIT 15
```

(Image 37) Schema of auto completion working across three layers of the system

Image 37 shows first part of process where every time user types in a character Ajax.Autocompleter equivalent of just simple observer sends it to controller.
Controller performs search on model where "LIKE" SQL is being executed for returning multiple elements that suite query. Limit at the end guarantees fast responses.
After some collection comes back controller builds up UL list with LI elements as pieces of collection from model. Whole thing is after that delivered into the view as a small dropdown menu where users can pick values from.

5.8. Caching

Caching allows significantly improve response times by providing pieces of data that were processed earlier and are now stored in some sort of temporary memory.
Caching can be performed at different levels of web application.

5.8.1 Views caching

When application is required to deliver content quickly we may need to consider delivering the most often views as static html called page caching. Rails by default support few types of caches for the view elements:

- Page Caching – The simplest to implement and use off all caching solutions in Rails. By simply putting *caches_page* directive followed by name of view in controller we have all RHTML, HAML, RJS, RXML view elements cached to corresponding static formats (HTML, XML, JS) As a result application server won't be involved in delivering these files – reverse proxy will do that. No need to process data, no DB query just simple html as in old days before dynamic pages come true.

- Action caching – Similar to page caching except it will pass through rails every time rather than just deliver static file. Page caching stores files inside public directory right in file system. Action caching is more flexible and can be told where to store files.

- Fragment Caching – This kind of caching in rails behaves very similar to action caching except it deals with partials (small elements of pages rather then whole views.)

The system uses Page caching mostly as other types of caching require that application is not getting results too dynamically, in business environment of cars sales where thousands of vehicles may change owners per week its not achievable. Elements cached as static pages are all static pages that won't change too often like all help pages, term of service and privacy policy.

Places where application uses caching permanently are JS arrays for searches drop downs.
They do not need to present data in real time and small lag guarantees fast delivery of static JS file containing all makes, models and possible years to search for.

5.8.2 Backend caching

Caching back ends, currently MoMo does not use that feature but idea is to have dedicated host, which has real time cache. Author's investigation led to memCache – C implemented infinite loop where small slices of data can be accessed nearly in real time. Perfect for rails as ruby has wrappers for memCache and lets models check if elements are in memCache before they proceed with executing SQL statement. Otherwise if cache does not have such an element after getting it from DB ActionRecord will deal with serialization and putting object into memCache. MemCache is planned for MoMo when it grows, adding it won't require any architectural changes. Currently system uses MySQL query caching for storing commonly accessed rows of data. For example earlier mentioned settings for application are accessed very often. Outcome of that is there is nearly no lag for getting them as InnoDB MySQL engine stores them in ram for quick access.

5.8.3 Sweeping caches

When using caching of views developer has to consider sweeping cache at some stage. Depending on application requirements it may happen once per week, day, hour or every 15 minutes. Thanks to earlier mentioned RAKE build tool MoMo has few scripts that clean caches by removing static files from hard drive preconfigured in a CRON jobs (Unix tasks scheduler).

Other way caches may be swept are the “observers” special models of type sweeper to purge caches on certain actions. Example when new car is added we could purge cache for all cars array in javascript. This seems like a good idea however sometimes changes happen too often and sweeping them per such a action would literally mean not caching at all.
That’s why MoMo uses own rake tasks for this on hourly basis.

6. MoMo now and in future

The author plans to deploy MoMo in the Irish market and possibly in the United Kingdom. After investigating different hosting options available it is certain that Engine Yard is the best for this system needs. This chapter summarizes the whole work, describes pros and cons of achieved results and plans for future. At the very end it gives some ideas about possible deployment scenarios.

6.1. Problems during MoMo creation

The biggest challenge during MoMo development was the definition of business requirements. To achieve that the author discussed the current sites and their features with many potential car buyers and dealers.
While tag search and most of the basic functionality were straightforward and easy to understand trade mode was not. To understand dealers' needs and to implement them correctly many models and prototypes must have been created and then evolved through tests. It was important to achieve balance between one time private sellers, dealers who make their living off selling cars to the public and dealers who trade solely to other dealers. The last group are not always interested in general public sales and focus their attention more on cars available in car trade market. So MoMo was designed with different types of sellers in mind with proved to be quite a challenge and let to develop trade mode features.

6.2. Pros & Cons of MoMo's current set of features

The main advantage of MoMo that everyone who seen the system mention is limitless way of finding car through different tags. People who had a chance to look into the system really appreciated the m-tag search.
Everybody mentioned this kind of functionality is superior to old ways – which it allowed them to nearly talk to interface what they wanted to search for.
Some people had hard time understanding those few new principles still the simple search and power search drop downs were well enough for their needs. What this group anticipated the most was browse that allowed them play with the system and browse back and forward without even realizing that they spent that much time on it.
The innovative bread crumbs turned out to be very useful and made people forget they have to use back button in their browsers to reach previously visited sites.
RSS feeds allow many dealers to view their own stock changes nearly in real time or to observe other dealers and cars for trade they may want to buy.
This is a big win for people with cars oriented business like different car sellers and dealers.

6.3. Ideas to introduce in future - possible improvements

Despite the fact the author is not big fan of thick clients. It may be possible to introduce one into MoMo in future – although it creates some problems with synchronization, versioning and distribution of different versions. There was still some small group of people among dealers that felt like client makes them own some part of the system. On top of that additional web services offering JSON or XML feeds for third party dealers sites are planed.

It is also hoped that the dealer in trade functionality can be extended to provide a more complete system to manage dealers only markets, which would involve a real time in trade and dealer reporting system. It is also hoped to add support for information about new vehicles as well as cars related news. Tagging could be extended further and depend more on users input.

6.4. Deployment on nginx and mongrel

For deployment needs the author tested few possibilities of deploying rails application. There are different roles to be performed by each element in the stack:

- Reverse proxy – delivering static content like javascripts, images or stylesheets.
- Application server – performs all the dynamic queries. (RoR MVC)
- Database server – for information storage.

Deployment possibilities are nearly infinite and as Rails were becoming more and more popular people were trying different ideas and solutions. New servers were born but only one or two are really production ready.
Mongrel [11] – written by Zed A. Shaw in C and wrapped with Ruby interfaces is easy to install (comes as a GEM) and to use – by default starts development environment of rails application on port 3000. Exactly the same as Webrick, which comes, bundled with rails. However Mongrel become number one app server among rails-ists as it is much faster and qps (query per second) rate is far more efficient than Webrick's.
Few people deploy rails as the FAST-CGI deployment – simple to use in an existing environment where Apache web servers are core of infrastructure. It is very popular among shared hosting companies. Still problem is that it is very hard to get it configured to work fast with rails. Most people complain about long interpretation period on first request and general slowness.
Apache is not uncommon choice as reverse proxy. As a part of every Unix like OS it is in most cases already installed and basically configured for this task. Nevertheless the author believes that Apache is too heavy and requires too many additional modules to perform different tasks that Nginx performs by default.
Nginx [10] has very small memory footprint just few kilobytes per worker process and about megabyte per main process. Additionally it compresses all static assets to gzip by defaults. Configuration of Nginx is fairly simple with big amount of help from people who already had experience and which can be easily found on the web.
Ezra Zygmuntowicz is the main Nginx supporter and has his company *Engine Yard* build on it – the MoMo uses his RoR deployment experience. [12]
Important thing about Nginx is that it does not lock app server connections till it is absolutely necessary. Proper example of this is file upload. While in normal deployment scenario mongrels/webricks are busy with process unable to serve another request here Nginx spawns another worker process which handles the upload and when file is physically on the server it is being passed for processing – no app server slot is taken till needed.

List Of Figures

Appendix A: Dictionary

AJAX – Asynchronous javascript and xml.

Car Portal/Car Wortal – Web Portal oriented on car sales. Wortal is a type of portal for specific subject in this example cars only.

Dealer – Someone who runs car sales business and is interested in multiple sales per week or even day.

Garage – Interface for sellers. Contains details about their stock, summaries and options for adding new cars.

GEM – Single plugin or library with some additional functionality that extends project.

GEM system – Set of ruby scripts for automatic downloading packages with new libraries for Ruby language. It is platform independent and based on idea behind apt-get Debian Linux package manager.

Mongrel – Application server used by MoMo

MVC – Pattern in software development for separation of different layers of application (Model, View, Controller)

Nginx – Reverse proxy/www server from Russia.

Private seller – MoMo defines private sellers as people who may sell their vehicle once for some period of time like every 2-3 years. It usualy means any private person eager to sale their current car.

Public Sale – Market for everyone to search and buy cars. Both dealers and private sellers have full access to it.

Reverse Proxy – Service, usually round robin server that maintains traffic distribution among application servers and is responsible for delivering static content.

RoR – Shortcut for Ruby on Rails web framework in ruby language.

RSS – Aggregation of information from website into XML webservice that can be read by special feed readers or processed by applications for further use. RSS is a part of every modern web 2.0 site.

Tag – A keyword attached to some text, object or data set which can be identified by it.

Trade Mode – Mode in MoMo to for dealers only market to exchange vehicles between themselves. Only dealer can put cars on that market and browse it.

Appendix B: References & Literature

[1]. Tantek Çelik , Kevin Marks - Microformats - "http://microformats.org/wiki/rel-tag" -

[2]. Ralf Wirdemann and Thomas Baustert – *"Restful principles"* - "http://www.b-simple.de/download/restful_rails_en.pdf"

[3]. Ezra Zygmuntowicz, Bruce Tate, Clinton Begin. *"Deploying Rails Applications"* Page 31 – Application issues for Deployment & Page 165 Nginx, from Russia with Love

[4]. Douglas K.Van Dueyne, James A. Landay, Jason I. Hong

"The Design Of Sites" page 121 clean interface

[5]. Dave Thomas, David Hansson, Leon Breedt, Mike Clark, James Duncan Davidson, Justin Gehtland, Andreas Schwarz. *"Agile Web Development with Rails, 2nd Edition"* – chapter about application flow and design.

[6]. Mike Clark *"Advanced Rails Recipes"* Page 25 – Toggle attributes with AJAX

[7]. Carzone website – „http://carzone.ie"

[8]. Cars Buyers Guide website – „http://cbg.ie"

[9]. Autotrader website – „http://autotrader.ie"

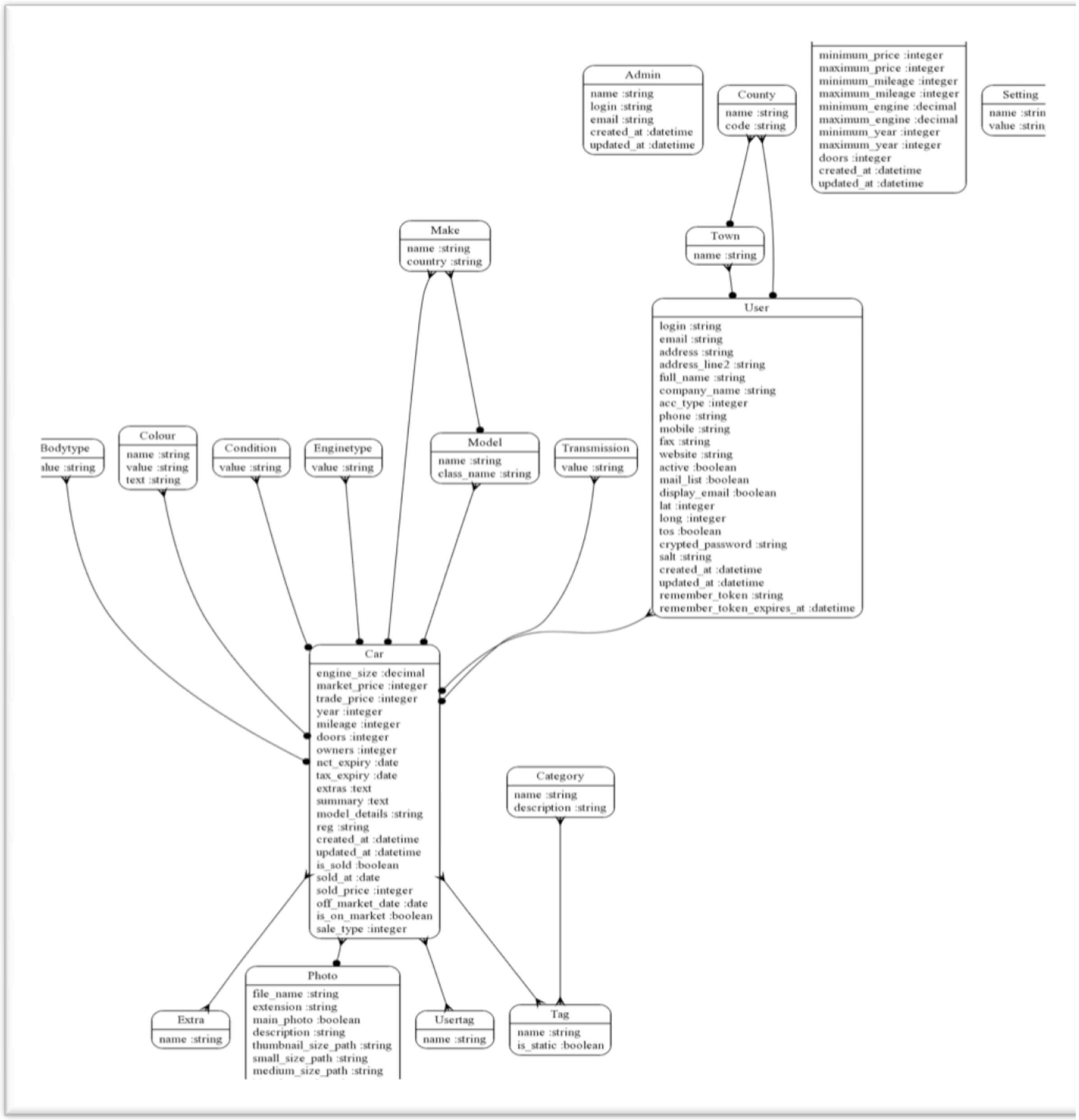
[10]. Igor Sysoev - Nginx reverse proxy – „http://nginx.net/"

[11]. Zed A. Shawn - Mongrel web/application server – „http://mongrel.rubyforge.org/"

[12]. Ezra Zygmuntowicz – „Xen and the art of rails deployment" - http://www.slideshare.net/vishnu/xen-and-the-art-of-rails-deployment

[13]. Delicious – bookmarking, social network site. *"http://del.icio.us"*

Appendix C: Schemas and Big Images



(Data base schema in MoMo)