

Advanced Web Technology

Rijksuniversiteit Groningen



Link Exchange Checker

Lecturer: Dr. A. Lazovik

Students: Zhe Sun

Jan Willem de Jonge

Alexander Bograd

2011/2012



Table of Contents

| | | |
|--------|-------------------------------------|----|
| 1. | Introduction..... | 3 |
| 2. | Context | 3 |
| 3. | Problem statement..... | 3 |
| 4. | Implementation and evaluation..... | 5 |
| 4.1 | Features..... | 5 |
| 4.1.1 | Twitter Bootstrap style UI | 5 |
| 4.1.2 | AJAX | 5 |
| 4.1.3 | Server Pusher (Web Sockets) | 5 |
| 4.1.4 | User feeling improving toolkit..... | 6 |
| 4.1.5 | Auto-suggestion | 6 |
| 4.1.6 | Maintaining of client session..... | 6 |
| 4.1.7 | Load balance..... | 7 |
| 4.1.8 | DB tolerant | 7 |
| 4.1.9 | Log | 7 |
| 4.1.10 | Complete error feedback | 8 |
| 4.2 | Functionality..... | 8 |
| 4.2.1 | Link checker | 8 |
| 4.2.2 | User management | 8 |
| 5 | Development environment | 9 |
| 5.1 | Platform..... | 9 |
| 5.2 | Tools and software | 9 |
| 5.3 | 3 rd party library | 9 |
| 6. | Technical architecture..... | 10 |
| 7. | Low level design | 11 |
| 7.1 | Link checker..... | 11 |
| 7.2 | User management..... | 12 |
| 7.3 | History Management..... | 13 |
| 8. | Conclusion | 14 |
| | References..... | 15 |



1. Introduction

The Link Exchange Checker (LECh) was developed as an assignment for the course Advanced Web Technology. This document describes the context, architecture, implementation and evaluation of the application.

2. Context

Internet started to grow rapidly in the mid-1990s. Since that time, the owners or administrators of web-sites began the process of optimizing those sites for search engines. Most of the techniques used in those years were heavy relying on tags and key words provided by the sites. The things have changed after Larry Page and Sergey Brin developed a search engine which relied on mathematical algorithm to rate importance of web-pages based on the quantity and quality of inbound links. Page and Brin founded Google in 1998. In the beginning it used the described algorithm to rank the web-pages. But many web-sites started to manipulate the search engine by creating link farms for just one reason, to make the sites rank higher. Since 1998 Google has changed its algorithm several times, therefore now the amount of incoming links is not as important as it was in the beginning.

3. Problem statement

Even though it is not as essential as it was before to know all the incoming links, there are still some cases where it could be important. For example, Alice and Bob both run their own web-sites. Their business is related, so they have agreed to exchange links. An agreement is OK, but how to make sure that this agreement is working? Web sites get more and more complex every day, plus Alice may be running several sites and has made agreements with many other web-site owners to exchange links. The Link Exchange Checker provides automated way to check for link exchanges.

The Link Exchange Checker is not the only tool on the market. There are many other tools that are very similar. Most of these tools are not free and they come as a part of the package with many other tools. But there are also some free tools that are very similar to The Link Exchange Checker, for example Lecheck.nl [1]

Lecheck.nl allows a quick way to check if the link partners still link back to the specified site. It has very similar functionality to our application, but its look is not as modern as our application.

Submitexpress [2] is another free tool that can check the back links, but the largest difference in this tool is that target URLs have to be entered manually.

Link-assistant [3] is a good example of licensed SEO suite, which is a client application with a lot of built in functionalities and one of them is a backlink checker.

As we have mentioned before, there are many tools to check for back links, but some of them are not free to use and the other ones lack some functionality which may be very handy for some of the users. Therefore, we propose a web-based application which will be deployed in the cloud, have some very interesting features, with possibility to sign up for the service and save the history results in the database.

The user interface of the application was somewhat inspired by the web-site <http://whatthefuckshouldimakefordinner.com/>, the screen shot of the application is in Figure 1.

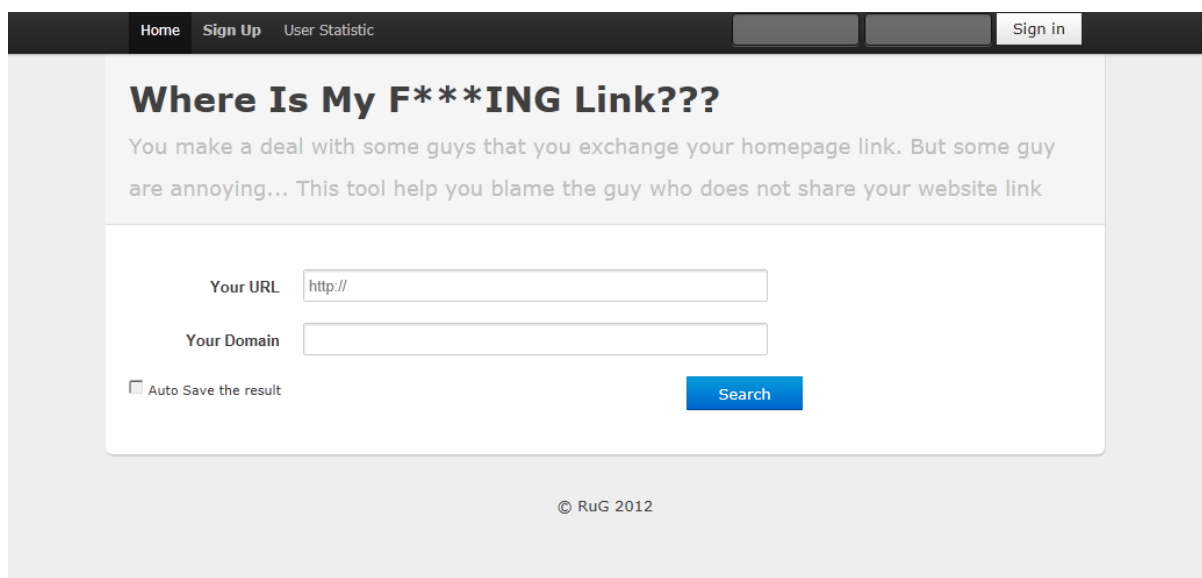


Figure 1. Main page



4. Implementation and evaluation

4.1 Features

4.1.1 Twitter Bootstrap style UI

Designing UI from scratch requires a lot of effort. Bootstrap [4] is an open-source front-end toolkit from Twitter that is aimed to help web-designers to kick start development of web applications and sites. Bootstrap is basically just a bunch of CSS, but it is built with Less [5], which provides more power and flexibility than regular CSS.

Bootstrap provides some basic templates for layout, typography, media, tables, forms, buttons, navigation, alerts & errors, popovers, and some other features.

4.1.2 AJAX

Ajax is widely used in the application. With its help we have an asynchronous web-application. We can send data and retrieve data from the server without interfering with the display of existing page. In our case we use it for search, signing up/in/out, fetching history records, etc.

4.1.3 Server Pusher (Web Sockets)

Searching for links is very time-consuming process, so we do not want the user to wait until the search is finished. We have to let them see the results as soon as a server has something. For that we use Web Sockets, which is a technology that provides bi-directional, full-duplex communication with a remote host over TCP connection. For easier implementation of Web Sockets, we have used Pusher [7], which is a simple hosted API that makes the use of Web Sockets very easy.

Web Sockets were introduced in HTML 5 specification.

We use this not only to show search results, but also for other information, like user statistic on the website.

Pusher has a simple Publish/Subscribe model based on channels that allows filtering and controlling how people receive messages. Pusher also supplies functionality such as authentication mechanisms for private channels, and presence functionality for keeping track of who's online. Figure 2 provides high-level view of Pusher architecture.

Understanding Pusher

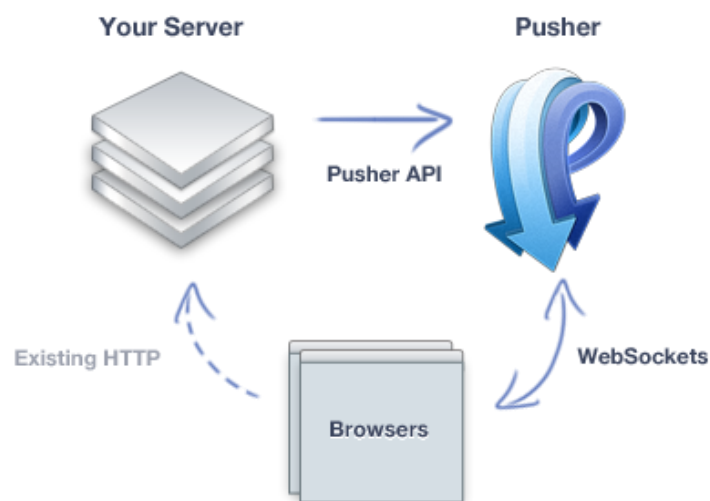


Figure2. Understanding Pusher

4.1.4 User feeling improving toolkit

Twipsy [6] is based on jQuery.tipsy plugin, which uses CSS3 for animation and data-attributes for title storage. We use it for auto-suggestion, so that the application looks really user-friendly.

jQuery Accordion plugin is used to expand/collapse content that is broken into logical sections, much like tabs. The underlying HTML markup is a series of headers (H3 tags) and content divs so the content is usable without JavaScript.

4.1.5 Auto-suggestion

In order to help users to enter the web-site and the domain, application uses auto-suggestion, which is based on the user history in case of entering the new web-site. And it suggests a domain based on the entered URL of the web-site.

4.1.6 Maintaining of client session

Maintaining of client session is very useful in some situations. We have also implemented it in our application. After a user logs in, the application creates a new session and maintains it. In case if the



user decides to open a new tab in the browser, he will be automatically logged in there. After the user logs out of application, the session will expire.

4.1.7 Load balance

Load balancing enables us to achieve greater fault tolerance of the application. For load balancing we are using Elastic Load Balancing provided by AWS. It distributes incoming application traffic across multiple Amazon EC2 instances. It also detects unhealthy instances within a pool and automatically reroutes traffic to healthy instances until the unhealthy instances have been restored.

4.1.8 DB tolerant

To make a fault tolerant database we implemented a MongoDB replica set with 3 servers. One of the servers will be the primary and the others are copies. If the primary fails any of the secondary nodes can take over, figure 3.

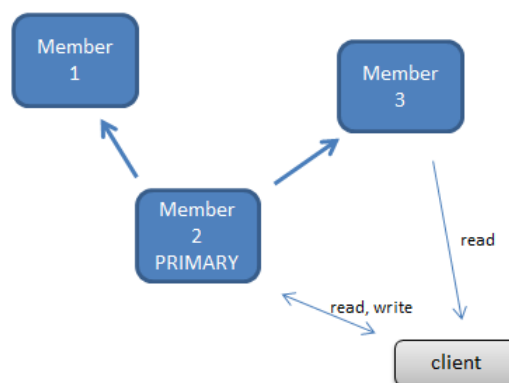


Figure3. Mongo replication.

4.1.9 Log

In order to get information about all the errors that may occur during run-time, we use log4j, which is an open source debugging tool, developed for putting log statements into the application.

A very interesting feature of log4j is Log Level. Users can choose 7 log and messages levels: off, fatal, error, warn, info, debug and trace. This way the developer can choose which level of log should be sent to the output stream.



4.1.10 Complete error feedback

In order to minimize user errors, we provide error checking in almost every step. It starts when the user wants to sign-up. We check if the user name already exists, if there was valid e-mail entered, minimum amount of symbols for the password, etc. Some things, like the confirmation of the password, happen on the fly, so the area in which the password has to be entered stays red until the right password was entered. Other validations, such as user name, happen only after a button was clicked.

For the validation of entered data, jQuery validation is used.

4.2 Functionality

4.2.1 Link checker

As we have mentioned in the previous chapters, the application checks for the exchanged links. There are two possible scenarios: the application can work with the user signed in or without signing in. The reason to sign in is to keep the history of the previous search results, which are stored in the database. Without signing in, it is also possible to use application, but it may be a while for the application to do a complete search, so it is not always handy.

All the results of the search are represented with the name of the page, URL and the status. Status has three possibilities: good link, bad link and unknown.

If the user is logged in, it is possible to save the search results in the history. There is also an option to delete the saved results.

4.2.2 User management

Every user is free to choose any user name. E-mail address is checked for the right structure, e.g. the existence of domain in the address. (com, net, etc.)

The password has to contain the minimum amount of symbols and it has to be entered twice to minimize the chance of making a mistake.

Every signed up user has an option to save all the search results in the history. There is also a possibility to save the search results to a file in CSV format. If there is no need to keep the results in the history, application provides functionality to delete not needed results.



5 Development environment

5.1 Platform

Amazon EC2 instance - is a web service that provides resizable compute capacity in the cloud. It is designed to make web-scale computing easier for developers.

Amazon load balance - automatically distributes incoming application traffic across multiple Amazon EC2 instances. It enables to achieve even greater fault tolerance in the application, seamlessly providing the amount of load balancing capacity needed in response to incoming application traffic. Elastic Load Balancing detects unhealthy instances within a pool and automatically reroutes traffic to healthy instances until the unhealthy instances have been restored [8].

5.2 Tools and software

The following tools and software was used in and for development of the application:

- Eclipse 3.6 – open source Java integrated development environment
- Java – a programming language
- Java SDK 1.6
- Javascript – a scripting language, which is used to provide enhanced user interface and make the web-site dynamic.
- HTML 5 – a language for structuring the content of the site. Even though the version 5 is still under development, it is still used in our application, because some of the used features, such as Web Sockets is not available in the previous versions.
- Chrome Developer Tools – it provides an integrated environment for debugging, optimizing and understanding a web application running in Google Chrome.
- MongoDB – open source document-oriented NoSQL database.

5.3 3rd party library

The following 3rd party libraries were used in the application. Some of them (Pusher, log4j) were described in previous sections.

- jQuery – a cross-browser JavaScript library. We have used it to simplify the client-side scripting of HTML



- Jsoup – an open source Java HTML parser, with DOM, CSS, and jquery-like methods for easy data extraction.
- JavaScript MD5 - MD5 is a secure hash algorithm and is used in the application to create a unique name for pusher channels.
- Table2CSV - this is small JQuery utility which allows exporting any HTML table as CSV file.
- Placeholder.js – HTML5 is still not fully supported by some of the browsers, so to make the application look the same in all the browsers, we have used this plugin.
- Gson – a Java library that is used to convert Java Objects into their JSON representations. It can also be used to convert JSON string to an equivalent Java object.

6. Technical architecture

The application was developed using a three-tier architecture, figure 4, which is very commonly used in web-development, especially the electronic commerce web-sites.

- A front end web-server serves the static and dynamic content, which is rendered by the browser. In our case we use Tomcat 7 as the web-server for the application.
- A middle layer processes the input which was made by the user. Java is used as a programming language for this layer.
- Database stores all the information which is needed for the user management and also it stores the history of the previous search results. MongoDB was chosen as a database.

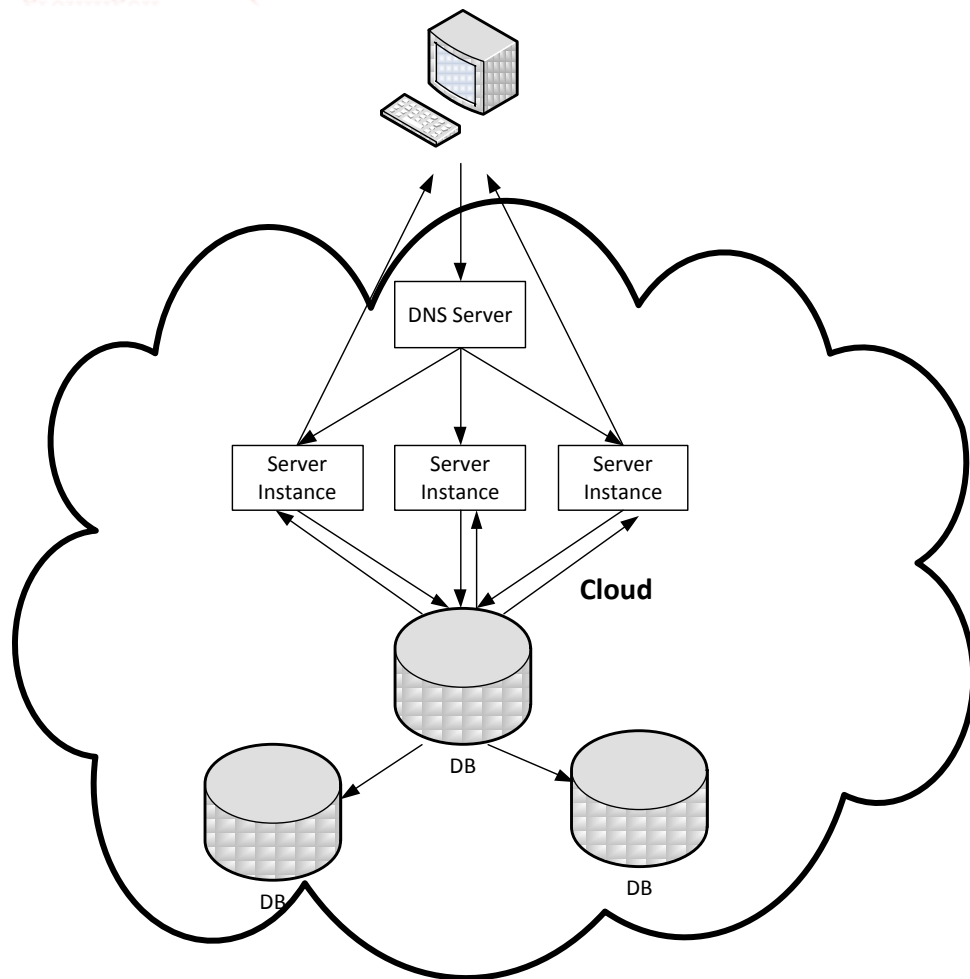


Figure4. Deployment diagram.

7. Low level design

7.1 Link checker

We have implemented the functionality of exchanging hyperlink checking by mainly using Jsoup and Pusher. Below is the procedure and some technical details:

1. Get the outgoing links.

When user submits the form, which inputs arguments including URL and domain, Jsoup fetches the URL, analyzes the HTML file and gets the URL string in attribution "href" in all "<a>" tags. If the URL string contains the domain string, that means the link points to a web page inside the user's site and it is not an outgoing link. The URL strings which do not contain domain string are the outgoing links.



2. Analyze the outgoing link in the backend server.

A thread is created to analyze all the outgoing links. Same as how we get the outgoing links, we use Jsoup to get the outgoing links of the outgoing-link website. If the outgoing link points to the URL the user submits, that website is clustered as a good site, otherwise it is a bad site.

3. Display the analyzed result in real time.

Sometimes the number of outgoing links could be very huge. The user does not have the patience to wait for a submission without response for a long time. To get a real time response, "Server Push" is the best choice. We choose Pusher, a flexible and well-structure solution. Pusher provides well-wrapped javascript method on the client side and java method on the server side. When user loads our website, a websocket channel is bound and listens to the messages from the server. After the websocket receives the message, a pre-bind event is triggered, i.e., a javascript function is called.

In our project, the web creates one Pusher channel permanently, which is used to receive the information of user statistics, i.e., the number of registered users, the number of online users and the number of visitors who are searching. Another channel, which receives link analyzing results, is created temporarily when a user submits a searching. Each analyzing result will be shown in the table of UI dynamically.

7.2 User management

User management uses a simple and clear structure. UI side takes charge in form submission; server side reads the information from DB, executes the action, then response UI.

To complete the User Management system, we use session management. Server side will create a new session when user signs in. The session will be destroyed in **HttpSessionListener**. In this way, when neither user signs out nor session expires because of no operation exceeding a specific time the current session will be terminated. Any UI action will first trigger a validation checking of session. This is very important for the modern browsers which have multiple tab. Suppose the scenario when the user opens two tabs in the browser to visits our website. The user signs out from one tab and then he wants to change his password in the second tab. We implemented the session validation checking for every servlet, that is to say, some action can only be executed during the lift cycle of a session, i.e., user status is sign in, and some action can only be executed when no user is signed in.



7.3 History Management

The user can save, delete and download the history. All the history information is stored in mongoDB. The database stores the information in the schema below:

- **SearchedID**

A unique search job identification string. It is composed of URL and domain.

- **LastVisitTime**

Show the latest visit time.

- **URL**

URL which user submits.

- **Domain**

Domain which user submits.

- **CntCouterpartUrl**

The number of outgoing links.

- **CntGoodLink, cntBadLink, cntTimeout and cntBadUrl**

These fields correspond the number of outgoing links which exchange our link, outgoing links which do not exchange the our link, outgoing links which can not fetch in a certain time (in our project it is 10 seconds) or is not reachable, outgoing links which are invalid URL.

- **Details**

List of outgoing links. Each element in the list corresponds to outgoing link and includes URL, text and status fields. "Text" means the text string in the anchor tag `Text`.

Saving and deleting are DB manipulations. Downloading is more complicated. We use the jQuery plugin "table2CSV.js" to generate a CSV format string from the table which stores the result. This CSV format string is then submitted to the server. The user gets the response which is the CSV file generated by the server.



8. Conclusion

Even though there are many applications that are very similar, our application has some advantages over it. It is really user-friendly, it uses the most advanced and modern technologies and it is scalable as well as fault tolerant.

The application was tested in different browsers: IE8, IE9, Google Chrome, Opera, Safari, Mozilla Firefox, and it looks and feels the same in all of them.

There is still some room for improvement and it may also be very handy to add some extra functionality, but we will leave it for the next version.



References

- [1] <http://www.lecheck.nl>
- [2] <http://www.submitexpress.com/>
- [3] <http://www.link-assistant.com>
- [4] <http://twitter.github.com/bootstrap/>
- [5] <http://lesscss.org/>
- [6] <http://twitter.github.com/bootstrap/javascript.html#twipsy>
- [7] <http://pusher.com>
- [8] <http://aws.amazon.com/elasticloadbalancing/>