Athens Institute for Education and Research
ATINER

ATINER's Conference Paper Series
COM2016-2063

Extending the Language of the Web for Dynamic Content Integration

Peter Stoehr
Professor
University of Applied Science Hof
Germany

Christin Seifert
Research Assistant
University of Passau
Germany

An Introduction to ATINER's Conference Paper Series

ATINER started to publish this conference papers series in 2012. It includes only the papers submitted for publication after they were presented at one of the conferences organized by our Institute every year. This paper has been peer reviewed by at least two academic members of ATINER.

Dr. Gregory T. Papanikos
President
Athens Institute for Education and Research

This paper should be cited as follows:

Stoehr, P. and Seifert, C. (2016). **"Extending the Language of the Web for Dynamic Content Integration",** Athens: ATINER'S Conference Paper Series, No: **COM2016-2063.**

Athens Institute for Education and Research
8 Valaoritou Street, Kolonaki, 10671 Athens, Greece
Tel: + 30 210 3634210 Fax: + 30 210 3634209 Email: info@atiner.gr URL: www.atiner.gr
URL Conference Papers Series: www.atiner.gr/papers.htm

ISSN: 2241-2891
24/11/2016

Extending the Language of the Web for Dynamic Content Integration

Peter Stoehr

Christin Seifert

Abstract

Dynamic web pages are usually displayed based on pre-defined rules for the dynamic orchestration of the content's sources. This requires to manually adapt the rules if the content sources change – a labour-intensive and cost-inefficient process. In this paper, we present an approach for semi-automatic integration of dynamic information into web pages. The integration of dynamic content is specified as queries for the underlying content database. We provide a prototypical solution, the SeCH-Browser (**S**elf **E**mbedding **C**haracteristic **H**yperlinks), implemented as an iOS application for Apple iPad tablets. The case study shows that parts of the content are created automatically at the time the page is rendered, which could help to simplify the process of web page maintenance.

Keywords: Content injection, Dynamic web pages, iOS application.

Acknowledgments: The presented work was in part developed within the EEXCESS project funded by the European Union Seventh Framework Programme FP7/2007-2013 under grant agreement number 600601.

Introduction

One of the major problems of web sites is to keep their information up-to-date. Since the manual maintenance of the content is a labor-intensive and tedious work, a process for automatically updating dynamic information is desirable. Web pages consist of three types of information:

- Static information, i.e., information that does not change over time. This includes historical data like birthdates, or company names.
- Semi-static information, i.e., information that does not change over a longer period of time. This includes for instance employee information on company web sites or recommendations for medical treatments.
- Dynamic information, i.e., information that has to be updated frequently to be up-to-date. Such information encompasses for example slideshows of museum artifact collections or lists of scientific publications for a specific topic.

Using content injection technologies can help to reduce the issue of keeping the dynamic information up-to-date.

Content injection is a technique used to include additional data into a web page at the time it is rendered. This leads to an automatic update of the content as soon as a visitor opens the web page.

We present an approach that builds on late binding content injection based on HTML micro format tags and gives an overview of the implementation of the application.

The remainder of the paper is structured as follows: In the next section we present the underlying idea of content injection. Section “Related Work” defines the requirements for the software to be used and evaluates several approaches within the scope of these requirements. Section “Approach” presents the underlying techniques of the SeCH-browser. Examples of the usage of the SeCH-browser are presented in section “Case Study”. An outlook on the next development steps finishes the paper.

Content Injection

Content injection approaches can be categorized into early binding and late binding content injection.

Early Binding Content Injection

The traditional way for content injection is to embed an HTML-link into a web page. Clicking on the link loads the content associated with it and displays it to the user.

A more advanced approach is based on database queries. Having well-defined dynamic data, e.g. data of a weather forecast, an application can make an automatic request to fetch the content and integrate it into the web page. However, this approach requires well-defined content and the

specification of the database request in some query language by the web designer.

These methods for adding additional content are based on information that may become outdated quickly, if not continuously checked for validity.

Thus, fixing the information related to the content description at the time of generating the web page can be described as an **early binding of information.** Obviously this kind of content injection shares the same problem as the HTML-content of a web page; the web page rapidly becomes outdated.

Late Binding Content Injection

To overcome the problems of content injection based on early binding our proposed solution is based on **late binding.**

Whereas early binding uses a predefined description of the content, e.g. the HTTP-address of the web page, late binding relies on a symbolic description to define the information that is going to be injected. While displaying the web page the symbolic information is used to dynamically integrate the additional information into the web page.

In contrast to early binding, in which all information is fixed by the time the author creates the web page, late binding defers the time of evaluating the information used for the content injection to the moment when the web page is displayed.

This paper describes an approach for late binding that is well suited for mobile devices with limited computational power and limited network bandwidth.

Related Work

In this section we first identify software requirements and then review related work with respect to the defined requirements.

The core of a late binding content injection system is the algorithmic part that is responsible for the extraction of the symbolic information that describes the data to be injected. This software has to fulfil the following requirements:

R1: As the information has to be extracted while the user is looking at the web page, soft real time behaviour of the algorithm is needed.
R2: The algorithm should work for every domain.
R3: The extracted information should be suitable to generate a query for a search engine, e.g. Google, Faroo, DuckDuckGo, independent of the search engine (the search engine is treated as a black box).
R4: As the application should be suitable for mobile devices with limited network capabilities the additional network traffic should be low.

Keyword Extraction Algorithms

Keyword extraction is the process used to identify a small amount of words that represent the core idea of a text element, e.g. a sentence, a paragraph or a complete document. The following selection of algorithms is based on the requirements defined above.

An early usage for keyword extraction techniques is the extraction of terms for tag clouds. Tag clouds appeared with the first Web 2.0 web pages and blogs and have been used to visualize the frequency distribution of the relevant keywords that describe the content of the web page. The first tag clouds used simple statistical analysis to detect the words with the highest frequency and used this information to generate a visual representation of the key words.

To improve the quality of the keywords, more sophisticated algorithms have been developed, e.g. the TextRank algorithm based on a random walk in a word-coocurrence graph (Mihalcea and Tarau, 2004). (Menaka and Rahda, 2013) provides an overview of keyword algorithms and compares their accuracy on different test data sets.

Preliminary tests with implementations of different algorithms (TextRank (Rose et al., 2010) and (Rada and Tarau, 2004), Repeated-String-Patterns (Tseng, 1998) and an algorithm proposed by (Barker and Cornacchia, 2000)) have shown that they fulfil the requirement R1 even on a mobile device like the iPad Pro. In addition these algorithms can be used for every domain, thus requirement R2 is fulfilled, too. In contrast to the results of (Seifert et al., 2013), where human readers where able to understand the core information of a document by using the generated key words, experimental tests that used keyword extraction algorithms to generate queries for standard search engines lead to the observation that the results are not sufficiently good. In most cases, the results were not specific enough for a relevant content injection. Thus, requirement R3 is not fulfilled.

Multi-Level Keyword Extraction

Schlötterer describes in (Schlötterer, 2015) a first approach that uses a multi level keyword extraction strategy. In an advanced system one part of the algorithm is based on an implementation of the dbpedia spotlight[1] algorithm running on a dedicated server (Schlötterer et al., 2014). Thus additional network traffic is generated and the requirement R4 is violated.

Additionally, tests using a Mac mini server, equipped with an 2,6 GHz Quad-Core Intel Core i7 Prozessor, showed that the computational power of mobile devices is currently not sufficient to ensure that the soft real time requirement can be fulfilled.

[1] http://spotlight.dbpedia.org/, last accessed June 24, 2016

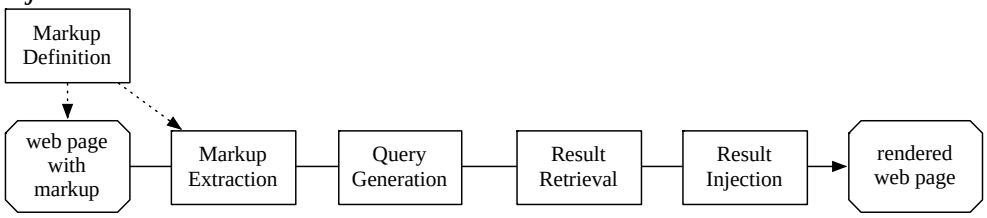
Approach

In this section we describe our query-based late binding approach to content injection. An overview of the approach is shown in Figure 1.

First, the micro format describing the content to be injected has to be defined (*Markup Definition*). A web page that is enriched with this information can then be processed by the web browser (*Markup Extraction*).

This symbolic information is used to create a query for a content provider (*Query Generation*), which is sent to the content provider to request the content for the late binding (*Result Retrieval*). Finally, the retrieved content is post-processed and integrated into the original web page before it is displayed to the user (*Result Injection*).

Figure 1. *Overview of the Query-based Late Binding Approach for Content Injection*



Markup Definition and Extraction

To overcome the problems described in the section above, the SeCH-web-browser uses HTML micro format tags to describe the symbolic information for the late binding content injection.

Currently three different HTML micro format tags[2] are used:

1. <meta name="search-head">, to describe the overall context of the web page.
2. <search-section>, to describe the context of a self-contained text entity.
3. <search-link>, to define the place and specific content of the content injection

At runtime, immediately before displaying the content of a page, the SeCH–web-browser extracts the information associated with the micro format tags and creates a query for every <search-link>. As a result, every time the user clicks on a <search-link> element the SeCH-web-browser displays a web page using these symbolic descriptions and generates an up-to-date content injection.

In contrast to fully automated content injections based on text-mining algorithms, where the website author loses control over the injected information, the SeCH–approach still allows the author to take care of the quality of the content injections.

[2] http://microformats.org/, last accessed June 24, 2016

Query Generation

The defined three tags describe the search context on different levels of granularity. The <search-link>-tag, as an anchor for the content injection, is the most specific description of the content, whereas the other two tags provide a more general description. These differences in granularity are also reflected in the representation of the query. The query is generated as follows:

search-link-info AND (search-section-info OR meta-info)

This ensures that information associated with the <search-link> tag is always included in the result set, combined with either the information associated with the <search-section>-tag, the <meta>-tag, or both of them.

Result Retrieval

For retrieving results, we query general purpose search engines (DuckDuckGo[3] and Faroo[4]) and a federated search service developed in the EU-project EEXCESS[5][6] aggregating results from specialized search engines like mendeley[7] or econbiz. The federated search is responsible for distributing the query to the registered and available content providers, retrieving their results, and finally merging and sorting the results. The federated approach has the advantages that i) content from different repositories can be queried without the necessity of an aggregated (and slow) content data base, and ii) still have access to a variety of data sources. The challenges for federation are the sorting and merging of results from various sources which differ in quality and vocabulary. Furthermore, the metadata format of all the sources has to be harmonized. The federated search returns a unified and aggregated result list, which is then rendered and injected into the web page.

Case Study

To show the feasibility of the approach, an iOS application for iPad systems and several web pages with embedded micro tags has been developed. The iOS application comprises the renderer for web pages, the markup extraction component, the component for generating and sending a query, and a component for displaying the retrieved results.

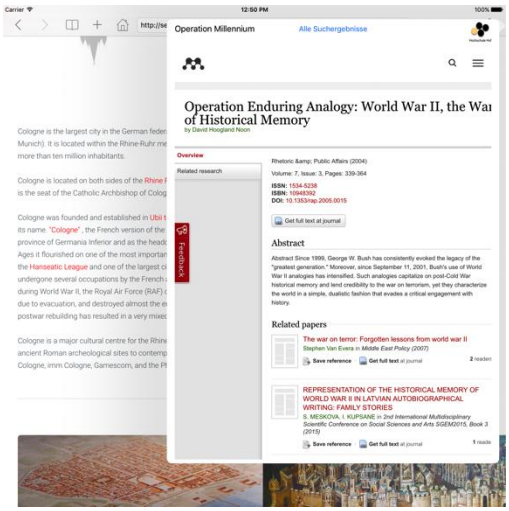
An example for the HTML source code with embedded tags is depicted in Figure 2. When selecting a <search-link>-tag the markup information is extracted; the query is built and sent to the federated search engine. Figure 3 shows the rendered web page from Figure 2 with the injected content overlaid.

[3] https://duckduckgo.com, last accessed June 23, 2016
[4] http://www.faroo.com, last accessed June 23, 2016
[5] http://eexcess.eu, last accessed June 23, 2016
[6] source code available from https://github.com/EEXCESS/recommender
[7] http://mendeley.com, last accessed June 23, 2016

Figure 2. *Web Page Source Code with Embedded Markup*

```
<head>
  …
    <meta name="search-head"
          topic="Cologne"
          type="location">
    <title>SeCH for Cologne</title>
  …
</head>
…
<search-section topic="World War II" type="misc">
    Cologne was one of the most heavily-bombed cities in Germany during World War II, the Royal Air Force (RAF) dropping
    <search-link topic="Operation Millennium">
      34,711 long tons of bombs
    </search-link>
    on the city.
    ….
```

Figure 3. *Web Page with Injected Content*



Summary and Future Work

In this paper we presented an approach for late binding content injection based on HTML micro formats. With a prototypical iOS implementation we showed that the solution is suitable for mobile devices with limited computational power and restricted network capabilities.

In the future, we aim to investigate the link-boosting and a re-ranking of the results from the search engines will be explored. Link-boosting will exploit the existing information associated with the HTML anchor tags to generate SeARCH-Links automatically at rendering time.

References

Barker, K., Cornacchia, N. 2000.Using Head Noun Phrases to Extract Document Keyphrases, In: Proceedings of the 13th Biennial Conference of the Canadian Society on Computational Studies of Intelligence: Advances in Artificial Intelligence, p. 40-52.

Menaka, S., Radha, N. 2013. An Overview of Techniques Used for Extracting Keywords from Documents. International Journal of Computer Trends and Technology (IJCTT). Volume 4 Issue 7.

Mihalcea, R., Tarau, P., 2004, TextRank: Bringing Order into Texts *Conference on Empirical Methods in Natural Language Processing.*

Rada, M., Tarau, P., 2004, TextRank: Bringing Order into Texts, Proceedings of EMNLP 2004, p. 404-411.

Rose, S., Engel, D., Cramer, N. and Cowley, W. 2010. Automatic Keyword Extraction from Individual Documents, In: Text Mining: Applications and Theory, Wiley, p. 3-20.

Schlötterer, J., 2015. From context to query, In: SAC '15: Proceedings of the 30th Annual ACM Symposium on Applied Computing, p. 1108-1109.

Schlötterer, J., Seifert, C., Granitzer, M., 2014. From Context-Aware to Context-Based: Mobile Just-In-Time Retrieval of Cultural Heritage Objects. In: Proc. Advances in Information Retrieval – European Conference on IR Research (ECIR) number 9022 in LNCS, p. 805-808.

Seifert, C., Ulbrich, E., Kern, R., Granitzer, M., 2013. Text Representation for Efficient Document Annotation, In: Journal of Universal Computer Science, Volume 19, number 3, p. 383-405.

Tseng, Y., 1998. Multilingual Keyword Extraction for Term Suggestion, In Proceedings of the 21st annual international ACM SIGIR conference on Research and Development in Information Retrieval, Melbourne, Australia, p. 377-378.