(X)querying RSS/Atom Feeds Extracted from News Web Sites: a Cocoon-based Portal

Giacomo Fiumara, Mario La Rosa, and Tommaso Pimpo

Dipartimento di Fisica, Università degli Studi di Messina
Salita Sperone 31, I-98166 Messina, Italy
giacomo.fiumara@unime.it

Abstract. The Web is fastly becoming the predominant source for news and information for many people. In the past few years, a new delivery system has emerged in the form of RSS feeds. Such feeds normally provide a brief of a larger news posted on the Web. RSS feeds, collected to form "channels" according to some thematic criteria, can be accessed using Web browsers or specialized software called "news aggregators". Even so, the amount of information available on the Web still exceeds human possibilities. In order to allow more selective and precise user choice, we developed a Web Cocoon-based platform which selects and publishes news gathered from various news Web sites. The selection is done submitting XQuery queries to a local repository and exploits the intrinsically semantic nature of RSS feeds.

1 Introduction and motivation

The World Wide Web (the Web) has become the predominant source for news and information for many people. To address the vast amount of content and the high frequency of news publication, a new delivery system has emerged in the form of "channels" or "feeds." These feeds, which are supplied by Websites such as CNN and BBC News, can be read using traditional Web browsers or specialized software, called "news aggregators." The two main formats for these feeds are RSS [4](Really Simple Syndication or Rich Site Summary) and Atom [16, 6]. They both provide an XML-based summary of the informational content of a website, with a brief description of the new "article" and links to the actual content. Feeds provide easy access to content in a pro-active mode, but presenting users with more content that they can handle. Current news aggregators do not provide the users very efficient means, beyond a simple keyword search, by selecting the most relevant content. Over a span of time, users will repetitively consider and discard content that does not match their interests. One major point is the impossibility to query, even in the relational sense of the term, the feed repositories before the retrieving of the data sources. An obvious advantage of such a "remote" query, would result in

a reduction of network traffic, less computational efforts and more pertinent content. We present here the preliminary results of our project, consisting in a Web site which publishes feeds retrieved by means of a series of queries (in Xquery[23, 12, 10] language) submitted to feed repositories spread across the Web. Registered users have the possibility to propose new repositories to be included in the set of those to be queried.

2 The software platform

The instruments we used for the development of our project (the graphical interface of the portal has been written in XHTML[21], Ajax[17] and CSS[19]) are entirely based on XML technologies; the development platform is the Apache Cocoon framework[18, 3, 8, 11, 14], which well suits for the construction of web applications by means of the aforesaid technologies. In the following we describe in some detail both Cocoon framework, XQuery and its potential, XSP programming language[26].

2.1 Cocoon

Cocoon was born as a Java servlet [20] with the aim of transforming XML documents through XSLT stylesheets[24, 25]. The community which coalesced around this project led it to its actual form, that is a Web-publishing framework built on the concepts of SoC (Separation of Content) [31, 32, 33, 34, 35] and component-based development of Web applications. Cocoon realized that mission by the notion of pipeline of components, where each component carries out a specific operation. Its creators define it "the web glue for your web application development needs", because SoC allows different development phases to coexist, thus reducing the possibilities of conflicts and error propagation.

Cocoon is based on the Avalon model[27, 28] and inherits its best features: first of all, the possibility of defining and developing new components. Components are defined by a descriptive interface and an implementation. For example, a parser is described by a Java interface that specifies all services it has to guarantee. Since this parser must be used inside an application, it's necessary for implementation to be conforming to the interface.

Cocoon's most important innovation is SoC-based design. During Web development, programmers often need to interfere with graphical designers' work and vice-versa, often resulting in a reduction of productivity. The purpose of Cocoon is to separate productive contexts to maximize the effectiveness of each team; style construction develops in parallel with logic design, improving productivity, quality and maintenability.

As per Web applications, the idea introduced by Cocoon is to use a pipeline to manage requests. A pipeline is a series of steps to process a particular kind of content. Usually, a Cocoon pipeline consists of a set of steps that

specify generation, transformation and serialization of SAX events composing generated content.

After being processed, requests move through pipeline stages. Each stage is responsible of a part of generation or transformation of contents. Cocoon allows to define all parts of a pipeline. SAX[30, 2] events are interposed between one phase and another; as an instance, the result of a pipeline can be a HTML page produced from a XML document.

A pipeline can be composed of four or more components always executed in the same order. For an example of pipeline see Fig. 1.

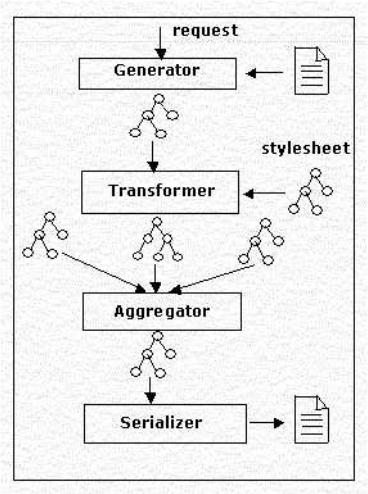

Fig. 1. A typical Apache Cocoon pipeline, from [18]

The sitemap is the heart of Cocoon. Here the developer configures Cocoon components and defines the client/server interactions in the pipeline. Cocoon matches each HTTP request to relative content in the sitemap, so that every part of the application (e.g. an XSLT file) is submitted to the appropriate component; each of them carries out a precise task and communicates with the precedent and/or the successive one by means of a stream of SAX events, activated when documents to be manipulated are submitted to the parser. SAX model consists of a set of classes and interfaces; it concerns two components placed in succession inside a pipeline; the first one sends a set of events, the following one pays attention waiting for these informations.

Transformations may be very demanding in terms of resources from servlet engine. Text parsing and transformations application require, in fact, a large quantity of processor resources. As to memory management, the situation has improved since Cocoon has adopted SAX in place of DOM[29], but this aspect is still problematic.

2.2 XQuery

XQuery is the language designed to query XML documents using XPath expressions. It's really a recent recommendation, become such through W3C on January 2007. It's not a fault to affirm that, from a semantic point of view, we are in front of a SQL for XML databases, as its aim is just this. XQuery syntax, however, is distanced from that one of its corresponding for relational databases: XQuery is, in fact, a procedural language made of functions (importable by means of namespaces), conditional and iterative instructions. The heart of language resides into **FLWOR** expressions, a set of five clauses (whose initials make the acronym) similar to that ones that form a SQL query:

- *For* assigns to a variable a list of elements, extracted from a XPath expression, involved in the XQuery query;
- *Let* operates a generic assignation (e.g. variable function value);
- *Where* establishes the condition to satisfy in the query;
- *Order* by establishes how results will be ordered;
- *Return* indicates the result of the query.

The argument of a clause is an expression in which function and XPath[22] expressions coexist. XQuery allows to embed code fragments inside HTML tags, on condition that they are delimited by braces. This feature permits to carry into effect, inside the same code and avoiding to recur to XSLT stylesheets, the separation between obtained data and their visual return.

2.3 XSP

XSP (eXtensible Server Page) is a language developed for Cocoon (by Cocoon developers) to create dynamic Web pages. It's still a technology under development, supported exclusively by this framework and composed of XML pages characterized by special tags. XSP programming is based essentially on three key points, through which separation between content and presentation is accomplished:

- use of tag libraries (logicsheets) imported by namespaces;
- use of a programming language (usually Java) inside appropriate markup elements;
- transformation of generated contents through XSLT stylesheets.

Each XSP page is processed by ServerPages generator, which represents in Cocoon the starting point of elaboration by means of pipelines. The ServerPagesGenerator transforms tags in a Java class which implements the Generator interface. XSP page is only compiled after first creation of the Generator; following executions will use the generator already available.

Each XSP pages starts with the $< xsp : page >$ tag; on its interior we declare the embedded programming language and the namespaces used to import tags from logicsheets. XSP supports programming language such

as Java, Javascript and Python. The rest of page comprises tags extracted from libraries and one or more $< xsp : logic >$ elements containing embedded code. XSP default library provides a further top-level element, called $< xsp : structure >$, in which declarations inherent to the used embedded language can be enclosed. Generally, it is used to declare the import of external modules as, for example, classes package. Being both logic and structure top-level elements, it's impossible to include one into the other.

Summarizing, an XSP page with only elements from default logicsheet introduces the following structure: a $< xsp : page >$ node, one or more $< xsp : structure >$ nodes, and one or more $< xsp : logic >$ nodes. The elements taken from this library don't allow a fluid XSP programming as they leave the development of dynamic content to embedded code, thus weighing down source code remarkably. Besides, it's advisable to divide code in syntactic markup blocks, each of them having its own function (session management, parameters management, etc.) and to commit what cannot be manipulated with these blocks to embedded logic or through creation of new specific logicsheets.

3 Our Project

The idea at the heart of our project [15] is to consider the Web as a huge database, each site representing an independent component which continuously generates updates. Thus, we face a multitude of information incessantly changing. It is also (more or less) homogeneously distributed on the whole network. Our goal is to retrieve RSS/Atom feeds published by some Web sites, store them in a Native XML Database (NXD) and publish them aggregated according to some filtering criteria, e.g. for thematic similarity. With respect to other Web-based feeds aggregators, we are able to submit XQuery queries to our repository, thus exploiting both the power of XQuery/XPath and the structure of RSS/Atom feeds. In order to publish feeds, we maintain a list of news sites which are frequently updated in order to retrieve fresh news. Our users can submit the URL of sites of her interest so to include them in our list. In order to enhance the performances of our portal, we decided to implement a caching mechanism, able to remember both the requested Web resources and the queries submitted by the users.

Indeed, each external URL access involves latency periods related to the nature of the connection. They grow linearly with the number of resources accessed.

A cache that memorizes the examined resource and the search parameters, has been used in order to eliminate this bottleneck. A deadline is assigned to each temporary version of the resources. It is defined as the parameter inside the pipeline, at the end of which the resource is considered stole. An additional Cocoon component (written in Java and inserted as a JAR) has been created in order to schedule cache updates. This "daemon" is like an

Action component inside the site-map. Moreover, the parameter concerning the duration of the cache is transferred.

3.1 Caching of Resources

Resources are served from an internal applicative pipeline which returns them through redirection, following a matching strategy studied for URLs that are corresponding to RSS and Atom files. This solution allows the storage and access to temporary copies of the requested resources, without causing modifications to the portal structure. Thus, together with the site-map, it defines an interface between our application and the Web.

A Java module has been implemented in order to schedule the access and then the storage of all resources in the cache through the creation of a connection and the request of an URL like http://www.feeding.it/allresources, which makes reference to a XQuery, forcing the update. During the first updating request a thread is created. It is kept in memory in order to satisfy the following updating requests and executed in parallel to both Java modules and searches. The response time is close to zero. If a search is executed during the updating operation, previously cached copies would be served.

3.2 Scalability Tests

A sequence of tests has been executed in order to study the speed of resources retrieval. The tests were made without using the cache, to better understand the updating times, with particular attention to differences of performance among searches inside and outside Italy. Each test has been made with 25, 50, 100, 200 resources and has been repeated ten times using the word "Iraq", first over Italian resources and then on non-Italian sites. Figures 2 and 3 illustrate the results of our test.

The max semi-dispersion here is represented by intervals of uncertainty enclosed within the upper and lower extremities which are respectively green and red. The rising of the resources coincides with the rising of intervals of semi-dispersion and a sub-linear growth of response times in the case of searches within Italy. We can note, moreover, how the response time for search done within Italy is extremely lower than that over the whole Web. This shows that latency times within the server determine performances.

3.3 Creating the portal

The portal has been called *Feeding* from the noun "feed" associated with the English suffix "ing" used to indicate action in progress, thus reflecting the nature of the project that handles data in continuous evolution. With the exception of thematic pages and search engine written in Xquery, the rest of the dynamic pages which compose the portal have been realized in

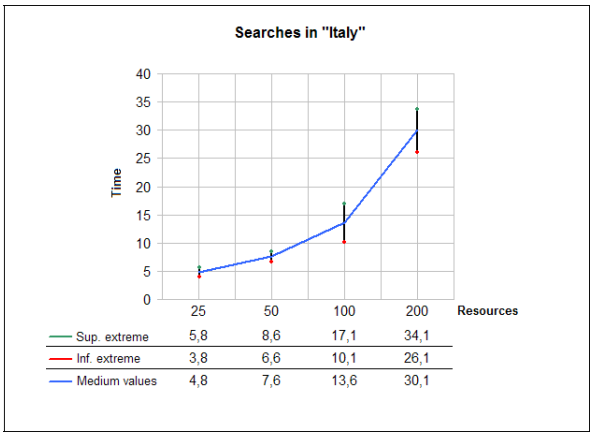

Fig. 2. Retrieval times vs no. of resources. Italian resources

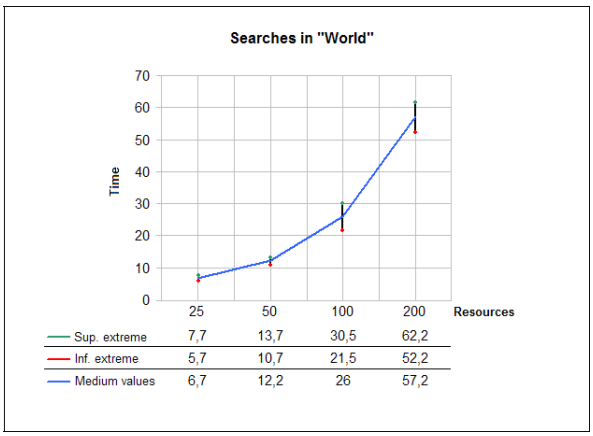

Fig. 3. Retrieval times of resources vs no. of resources. Non Italian resources

XSP and acquire the respective contents through SQL query on HSQL-DB, a RDBMS integrated in Cocoon. Queries are embedded in the XSP tags, then transformed through XSLT and XSL-FO style sheets. The native XML-DB, Exist, offers the Xquery support used to query RSS/Atom feeds. The graphic interface is written in CSS 2.0, XHTML-Transitional 1.0 and AJAX.

3.4 Creating the search engine

The search engine has been entirely written in Xquery. *Feeding* uses an XML file with the URLs of each feeds. Two attributes, which indicate the "language" and the "topic", have been assigned to each URL, respectively. When search is done, the selected key-words and the radio-button index are transmitted by URL-rewriting to the page of interest. The news source acts as a

filter in order to select the XML-path used in the query. It is in fact the result of a Xquery function that uses the above parameters as arguments. The main function of the engine uses the search keys obtained and returns all the occurrences within the elements *item/title* and *item/description* of the feed. This procedure is iterated for all RSS/Atom resources of interest. The news of each feed are then listed ordered by publication date.

Feeding allows the use of advanced functions in order to obtain a highly selective search. Selection criteria may be specified in one of these forms:

- Basic search: *Università Messina*. Looks for the first OR the second word occurrence in the feed.
- Pattern search: *"Università di Messina"*. Looks for the exact pattern occurrence in the feed.
- Exclusion search: *Università -Messina*. Looks for Università without Messina matching.
- Inclusive search: *Università +Messina*. Looks for both Università AND Messina, meaning that the feed has to have both words at once.
- Search with date: *2007-04-07*. Looks for all news published in the specified data. The data has to be written in the form YYYY-MM-DD, MM and DD can be optionally excluded.

3.5 Complex searches

One or more of the listed search forms can be used at the same time, thus allowing the users to make complex searches as: *Università -Messina 2007-03-07*. It looks for all news published in the specified data with the matching word "Universitá" and without "Messina". Even better, a user can search for terms appearing in the *title* field, other terms appearing in the *description* field while limiting the feeds only to those published within a time period, say a couple of days.

3.6 The thematic pages

The content of each thematic page is generated through a query which acquires its parameters through URL rewriting and uses it to select the resources from which the news will be extracted. We notice that each URL inside the XML file is equipped with an attribute that specifies the topic of the feed. The content of the page is generated through a query which associates the acquired parameter during the request according to the value of the attribute mentioned above. The result of the query includes the latest news published for each feed.

4 Related work

Apache Cocoon is a successful framework and by now it has been deployed at several sites[1]; some of which exploit its main feature, that is the separation of content, logic and presentation. In the few last years also some scientific projects adopted Apache Cocoon as a framework for their applications, even if their field of interest differs from ours. See [1, 9] for sample applications.

As to the main goal of our project, that is management of repositories of RSS/Atom feeds and the subsequent extraction of relevant information, we found a correspondence in the works on information extraction tools. These, whose aim is to convert semi-structured or structured Web content into a structured, i.e. XML, format, have been thoroughly surveyed from a number of authors. See for example [36, 37, 38] and references therein.

5 Conclusions and future work

We presented a new platform for retrieval and querying of RSS/Atom feeds by means of a powerful XQuery engine, which fully exploits the structure of XML documents. Selected RSS/Atom news sites are frequently queried and newly produced feeds are retrieved and stored in a local XML database for future queries. Although our project is still in a early development stage, its first results seem promising and the emphasis on Xquery queries are unique among various feeds portal on the Web. We planned, as our next achievings, to better manage feeds polling to minimize the number of unnecessary feed retrievals and to publish our platform on the Web.

References

1. Eidenberger H (2004) Modelling of Visual Feature Derivation in the Vizir Framework. Proceedings European Signal Processing Conference, Vienna
2. Faragas L (2004) The Joy of SAX. First International Workshop on XQuery Implementation, Experience and Perspectives, Paris, France
3. Ford N (2003) Art of Java Web Development: Struts, Tapestry, Commons, Velocity, JUnit, Axis, Cocoon, InternetBeans, WebWork. Manning Publications
4. Hammersley B (2005) Developing Feeds with RSS and Atom. O'Reilly Media, Inc.; 1 edition
5. Jafari A (2003) Designing Portals: Opportunities and Challenges. Information Science Publishing
6. Johnson D (2006) RSS and Atom in Action: Web 2.0 Building Blocks. Manning Publications
7. Kraus A, Koch N (2002) Generation of Web Application from UML Models using an XML Publishing Framework. 6th World Conference on Integrated Design and Process Technology, Pasadena, CA

[1] See http://cocoon.apache.org/link/ for an updated list

8. Leung T W (2003) Professional XML Development with Apache Tools: Xerces, Xalan, FOP, Cocoon, Axis, Xindice. Wrox
9. Madeyski L, Stochmialek M (2004) Architecture of Modern Web Application. Software Engineering after the year
10. Melton J, Buxton S (2006) Querying XML: XQuery, XPath, and SQL/XML in context. Morgan Kaufmann
11. Moczar L, Aston J (2002) Cocoon Developer's Handbook. Sams; 1st edition
12. Robie J (2003) SQL/XML, XQuery, and Native XML Programming Languages. XML Conference and Exposition, Pennsylvania Convention Center, Philadelphia, PA
13. Sangmi L, Sunghoon K, Fox G (2003) Adapting Content for Mobile Devices in Heterogeneous Collaboration Environments. ICWN Cocoon
14. Ziegeler C, Langham M (2002) Cocoon: Building XML Applications. Sams; Pap/Cdr edition
15. La Rosa M, Pimpo T (2007) Ricerca di feeds RSS/Atom su database dinamici distribuiti: un portale con il framework Cocoon. Graduation project. University of Messina
16. Wittenbrink H (2005) Rss And Atom: Understanding And Implementing Content Feeds And Syndication. Packt Publishing
17. Garrett J J (2005) Ajax: A New Approach to Web Applications. http://www.adaptivepath.com/publications/essays/archives/000385.php
18. Apache Cocoon Project http://cocoon.apache.org
19. W3C CSS 2.1 Specs http://www.w3.org/Style/CSS/
20. Sun Java Enterprise Edition http://java.sun.com/javaee/
21. W3C XHTML 1.0 Specs http://www.w3.org/TR/xhtml1/
22. W3C XPath Specs http://www.w3.org/TR/xpath
23. W3C XQuery 1.1 Specs http://www.w3.org/XML/Query/
24. W3C XSL http://www.w3.org/Style/XSL/
25. W3C Xslt http://www.w3.org/TR/xslt
26. Apache Cocoon Project - XSP http://cocoon.apache.org/2.1/userdocs/xsp.html
27. Apache Avalon model http://cocoon.apache.org/2.1/developing/avalon.html
28. Apache Excalibur Project http://excalibur.apache.org/
29. W3C DOM http://www.w3.org/DOM/
30. SAX Project http://www.saxproject.org/
31. Hursch L, Videira Lopes C (1995) Separation of Concerns. TR NU-CCS-95-03, College of Computer Science, Northeastern University, Boston, MA
32. Kener C, Kirda E (2000) Layout, Content and Logic Separation in Web Engineering. 9th International WWW Conference, 3rd Web Engineering Workshop, Amsterdam
33. Burner A (2002) Comparison of Web Technologies and Web Engineering Methodologies. BurnerNet.com
34. Reina A M, Torres J, Toro M (2003) Aspect-Oriented Web Development vs. Non Aspect-Oriented Web Development. Workshop of analysis of Aspect-Oriented Software, Darmstadt, Germany
35. Aksit M (1996) Composition and Separation of Concerns in the Object-Oriented Model. ACM Computing Surveys
36. Laender A.H.F. , Ribeiro-Neto B.A., da Silva A.S., Teixeira J.S. (2002) A Brief Survey of Web Data Extraction Tools SIGMOD Records 31(2) 2002
37. Flesca S., Manco G., Masciari E., Rende E. and Tagarelli A. (2004) Web wrapper induction: a brief survey. AI Communications 17 (2004) 57 - 61

38. Chia-Hui Chang, Kayed M., Girgis M.R., Shaalan K. (2006) A Survey of Web Information Extraction Systems IEEE Transactions on Knowledge and Data Engineering, TKDE-0475-1104.R3