The Personal Publication Reader: Illustrating Web Data Extraction, Personalization and Reasoning for the Semantic Web

Robert Baumgartner[1], Nicola Henze[2], and Marcus Herzog[1]

[1] DBAI, Institute of Information Systems,
Vienna University of Technology,
Favoritenstrasse 9-11, 1040 Vienna, Austria
{baumgart, herzog}@dbai.tuwien.ac.at
[2] ISI - Semantic Web Group, University of Hannover,
Appelstr. 4, D-30167 Hannover, Germany
henze@kbs.uni-hannover.de

Abstract. This paper shows how Semantic Web technologies enable the design and implementation of advanced, personalized information systems. We demonstrate by means of an example application how personalized content syndication can be realized in the Semantic Web. Our approach consists of two main parts: The web data extraction part, providing the information system with real-time, dynamic data, and the personalization part, which deduces - with the aid of ontological domain knowledge - personalized views on the data. The prototype of the system has been realized using the *Personal Reader Framework* for designing, implementing, and maintaining Web content Readers[1].

Keywords: semantic web, personalization, reasoning on the semantic web, web data extraction.

1 Motivation

The realization of the Semantic Web idea to be "an extension of the current web in which information is given a well-defined meaning, better enabling computers and people to work in cooperation" [5] has in only a few years pushed researchers and computer specialists to explore machine-readable semantics, appropriate markup and description languages, and sharable knowledge representation techniques. While these before mentioned techniques exist (at the writing time of this paper) as W3C recommendations, is the design of the so-called upper layers of the Semantic Web tower[4], e.g. the rule and reasoning layer, or the layers of proof and trust, still to explore.

[1] This research has been partially supported by REWERSE - Reasoning on the Web (rewerse.net), Network of Excellence, 6th European Framework Program.

A. Gómez-Pérez and J. Euzenat (Eds.): ESWC 2005, LNCS 3532, pp. 515–530, 2005.

In this paper, we investigate how advanced information systems for the Semantic Web can be realized. We claim that a huge class of Semantic Web-enabled information systems should be able to extract relevant information from the web, and to process and combine pieces of distributed information in such a way that the content selection and presentation fits to the current and individual needs of the user. From this viewpoint, such systems need to focus especially on the *information extraction process*, and the *personalized content syndication process*. The actual authoring process of information, and the information management processes, are important aspects, too, if we consider portal-like applications. However, there is a sustainable need of systems which can detect and process already existing Web information. To demonstrate our ideas for personalized content syndication, we consider the following scenario:

> Peter is working as a researcher at a university. He publishes his research findings in journals and conferences, and also puts his publication online onto his institute's homepage. Peter is also enrolled in a research project. From time to time, he is requested to notify the project coordination office about his new publications.
> The project coordination office maintains a member page where information about the members, their involvement in the project, research experience, research publications, etc. is maintained.

When we analyze the scenario, we see that

1. data about the publications is duplicated - it is stored at the university where Peter is working, but also on the Web pages of the project,
2. information about the project (people, research goals, achievements, etc.) is available online, but not related to the publications (unless somebody relates this information by hand).

The questions at hand from the scenario are:

- Can we organize this process in a way that Peter needs to publish his publications only once, e.g. at his institute's Web page? Thus that we avoid duplication of information, together with all negative side-effects like maintenance and update problems?
- Can we make use of the available contextual information on the project?
- Can we extract (relevant) information from Web pages?
- Can we combine the data in an intelligent way in order to provide a user a personally optimized access to the information? From the scenario, we may conclude that information about the role of researchers in the project like "Bob is participating mainly in working group X, and working group X is about topics Y and Z. strongly cooperating with working groups Y and Z" might be available. If we succeed in making this information available to machines to reason about, we can derive new information like: "This research paper of Bob is related to working group X, other papers of working group X on similar research questions are A, B, and C, etc."

This paper answers the above stated questions and demonstrates their realization within the Personal Reader framework[8, 9]. We have implemented a Personal Reader instance, the so-called *Personal Publication Reader (PPR)* which makes use of web data extraction techniques, reasoning about ontological knowledge and metadata description of informations, and provides a personal semantic view on publication data. The Personal Publication Reader has been designed and developed in the context of the Network of Excellence "REWERSE - Reasoning on the Web" and syndicates and personalizes information about the project structure, people and objectives of the REWERSE project, etc., and information about research papers in the context of the project.

The paper is organized as follows: In Section 2 we briefly outline our idea of establishing personalization services for the Semantic Web, and describe the architecture of the Personal Reader framework. The following Section 3 discusses approaches for Web Data extraction, and introduces the Lixto Suite. Section 4 then describes the realization of the Personal Publication Reader (PPR) in detail: We describe what kind of data is available via the Web (Section 4.1), and how we extract (Section 4.2), and transform it (Section 4.3) for the PPR. The domain ontology of the PPR, describing the REWERSE project, its members and research objectives, is topic of Section 4.4. Section 4.5 shows how various personalization rules derive new facts as well as personalized views on the data on top of extracted data, ontological knowledge, and user profile information. Concluding remarks and an outlook on ongoing and future work end this paper.

2 Personal Web Content Readers

Flexible information systems which need to be capable of adjusting to different application domains require a different architecture: not a monolithic approach, but several, independent components, each one serving a specific purpose. The recent Web service-technology focuses on such-like requirements: A Web service encapsulates a specific functionality, and communicates with other services or software components via interface components (e.g. [20, 15]).

We consider each (personalized) information provision task as the result of a particular service (which itself might be composed of several services, too). The aim of this approach is to construct a Plug & Play - like environment, in which the user can select and combine the kinds of information delivery services he or she prefers. With the Personal Reader Framework, we have developed an environment for designing, implementing and maintaining personal Web content Readers [8, 9]. These personal Web content Readers allow a user to browse information (the *Reader* part), and to access personal recommendations and contextual information on the currently regarded Web resource (the *Personal* part). The next section outlines briefly the architecture of the Personal Reader framework.

2.1 The Personal Reader Framework: Designing and Maintaining Personal Web Content Readers

The architecture of the Personal Reader framework is a rigorous approach for applying Semantic Web technologies. A modular framework of Web services – for constructing *the user interface*, for *mediating* between user requests and currently available personalization services, for *user modeling*, and for offering *personalization functionality* – forms the basis of each Personal Reader Instance (see Figure 1).

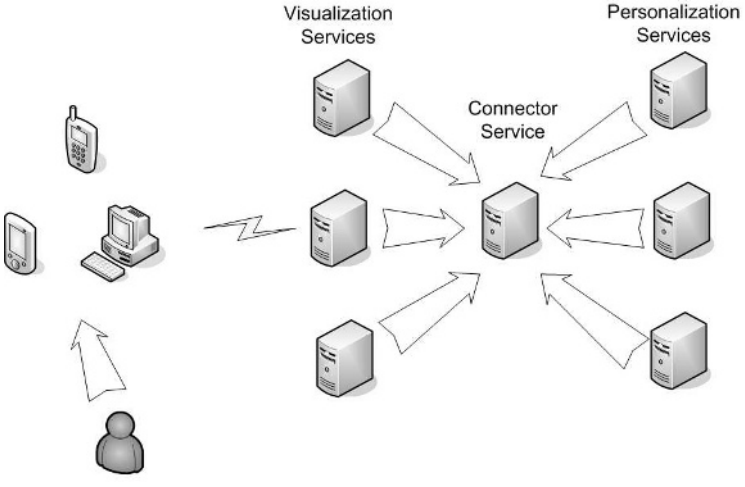

Fig. 1. Architecture of the Personal Reader framework, showing the different components of the Personal Reader: visualization, personalization, and the Personal Reader backbone (consisting of the connector service which organizes the communication and matching between the various visualization and personalization services)

The aim of the Personal Reader framework is to realize Web content Readers which give the user the possibility to select services, which provide different or extended functionality, e.g. different visualization or personalization services, and combine them into a *personal* Web content Reader instance. The framework features a distributed open architecture designed to be easily extensible. It utilizes standards such as XML[21], RDF[17], etc., and technologies like Java Server Pages (JSP)[11] and XML-based-RPC[22]. The communications between all components / services is syntactically based on RDF descriptions. This provides the required flexibility for combining various personalization and visualization services in one application, and thus supports the realization of our Plug & Play idea for personalization functionality on the Semantic Web.

2.2 Related Work on Personalized Information Systems

To the best of our knowledge, we are not aware of personalized information systems on the Semantic Web which realize the *personalization-as-service* idea in a

similar way. Personalized information systems require a sophisticated model of the actual application domain, thus, traditionally, these systems do not provide (and do not aim for) extensible architectures and systems. However, in [10], we have conducted a study on the re-usability aspects of personalization functionality, with special focus on the area of adaptive hypermedia systems. This study led to the conclusion that in fact even highly system-dependent personalization functionality like those from adaptive hypermedia research, can be encapsulated and prepared for re-use, an important precondition for the successful realization of personalization services is given.

3 Web Data Extraction and Integration

3.1 Objectives and Approaches

The *unstructured Web* of today contains millions of documents which are not query-able as a database and heavily mix layout and structure. Moreover, they are not annotated at all. There is a huge gap between Web information and the qualified, structured data as usually required in corporate information systems or as envisioned by the Semantic Web. However, until the vision of a Semantic Web is realized, and also, towards a faster achievement of this goal, it is absolutely necessary to (semi-)automatically extract relevant data from HTML document and automatically translate this data into a structured format, e.g., XML. Once transformed, data can be used by applications, stored into databases or populate ontologies.

A program that automatically extracts data and transforms it into another format or markups the content with semantic information is usually referred to as *wrapper*. Wrappers bridge the gap between unstructured information on the Web and structured databases. A number of classification taxonomies for wrapper development languages and environments have been introduced in various survey papers [6, 12, 13]. In general, it is distinguished between high-level programming languages, machine learning approaches and supervised approaches. Due to the lack of space we refer to the mentioned survey papers for an overview of available methods and tools.

3.2 Lixto Visual Wrapper

Lixto Visual Wrapper [2] is a methodology and tool for visual and interactive wrapper generation developed at the University of Technology in Vienna together with the Lixto Software GmbH. It allows wrapper designers to create so-called "XML companions" to HTML pages in a supervised way. As internal language, Lixto relies on *Elog*. Elog is a datalog-like language especially designed for wrapper generation. The Elog language operates on Web objects, that are HTML elements, lists of HTML elements, and strings. Elog rules can be specified fully visually without knowledge of the Elog language. Web objects can be identified based on internal, contextual, and range conditions and are extracted as so-called "pattern instances".

In [7], the expressive power of a kernel fragment of *Elog* has been studied, and it has been shown that this fragment captures monadic second order logic, hence is very expressive while at the same time easy to use due to visual specification.

Besides expressiveness of a wrapping language, robustness is one of the most important criteria. Information on frequently changing Web pages needs to be correctly discovered, even if e.g. a banner is introduced. Visual Wrapper offers robust mechanisms of data extraction based on the two paradigms of tree and string extraction. Moreover, it is possible to navigate to further documents during the wrapping process. Validation alerts can be imposed that give warnings in case user-defined criteria are no longer satisfied on a page.

The usage of Elog is completely invisible to the average wrapper designer and all operations are carried out by visual means. This is comprised of two steps: First, the identification phase, where relevant fragments of Web pages are extracted (see Figure 2). Such extraction rules are semi-automatically and visually specified by a wrapper designer in an iterative approach. This step is succeeded by the structuring phase, where the extracted data is mapped to some destination format, e.g. enriching it with XML tags to subsequently populate an ontology with instance data.

3.3 Lixto Transformation Server

Heterogeneous environments such as integration and mediation systems require a conceptual information flow model. The usual setting for the creation of services based on Web wrappers is that information is obtained from multiple wrapped sources and has to be integrated; often source sites have to be monitored for changes, and changed information has to be automatically extracted and processed. Thus, push-based information systems architectures in which wrappers are connected to pipelines of post-processors and integration engines which process streams of data are a natural scenario, which is supported by the Lixto Transformation Server [3]. The overall task of information processing is composed into stages that can be used as building blocks for assembling an information processing pipeline. The stages are to

- acquire the required content from the source locations; this component resembles the Lixto Visual Wrapper plus Deep Web Navigation and Form iteration;
- integrate and transform content from a number of input channels and tasks such as finding differences,
- interact with external processes, and
- format and deliver results in various formats and channels and connectivity to other systems.

The actual data flow within the Transformation Server is realized by handing over XML documents. Each stage within the Transformation Server accepts XML documents (except for the wrapper component, which accepts HTML), performs its specific task (most components support visual generation of mappings), and produces an XML document as result. This result is put to the

successor components. Boundary components have the ability to activate themselves according to a user-specified strategy and trigger the information processing on behalf of the user. From an architectural point of view, the Lixto Transformation Server may be conceived as a container-like environment of visually configured information agents. The pipe flow can model very complex unidirectional information flows (see Figure 3). Information services may be controlled and customized from outside of the server environment by various types of communication media such as Web services. The Transformation Server includes a user management that allows application designers to subscribe and parameterize components of other application designers.

4 The Personal Publication Reader

To realize the Personal Publication Reader (PPR) within the Personal Reader framework (see Section 2), we extract the publication information from the various Web sites of the partners in the REWERSE project: All Web pages containing information about publications of the REWERSE network (see Section 4.1) are periodically crawled and new information is automatically detected, extracted and indexed in the repository of semantic descriptions of the REWERSE network (see Sections 4.2, 3.3, 4.3). Information on the project REWERSE, on people involved in the project, their research interests, and on the project organization, is modeled in an ontology for REWERSE (see Section 4.4). Extracted information and ontological knowledge are used to derive a syndicated view on each publication: who has authored it, which research groups are related to this kind of research, which publications are published by the research group, which publications are on the similar research, etc. Information about the current user of the system (such as specific interests of the user, or his membership to the project) is used to individualize the view on the data (see Section 4.5).

4.1 Publication Data on the Web

In this scenario we are in particular interested to give a personalized view on publications of the members of the REWERSE network of excellence. Therefore, the ontology of the Personal Publication Reader has to be populated with instance data from publication sources. In most of the cases, the organizations offer access to their publications through a Web interface. However, each Web presentation is totally different, some use e.g. automatic conversions of *bibtex* or other files, some are manually maintained, some are based on databases. Such a presentation is well suited for human consumption, but hardly usable for automatic processing. Nevertheless, the Web is the most valuable information resource in this scenario. In order to access and understand these heterogeneous information sources one has to apply web extraction techniques as described in Section 3.

In Table 1 selected REWERSE members are given and their publication format is described. The table explains how the publications are structured, and

Table 1. Publication Web pages of selected REWERSE members

Participant	*Structure and Presentation*
Munich	`http://www.pms.informatik.uni-muenchen.de/publikationen` all publications on a single page sorted by years (latest on top), auto-generated format, usage of HTML elements inside publications, even for individual authors, links and bibtex available
Hannover	`http://www.kbs.uni-hannover.de/Stamm/Publikationen.html` all publications on a single page sorted by years (newest on top, publications numbered by years), publications consistent (some formatted differently), data very complete, usage of HTML elements inside publications, links available
Heraklion	`http://www.ics.forth.gr/publications.jsp` publications on multiple pages structured by years; additional structuring with next links, sites and publications consistent, data very complete, usage of HTML elements inside publications, links and abstracts available
Linköpping	`http://www.ida.liu.se/ext/dpr/access2/` publications on multiple pages structured by years, sites and publications not consistent, usage of HTML elements inside publications, links on selected authors, publications numbered

how the format of a single publication looks like. Moreover, it describes whether at least some parts of a single publication are rendered via HTML elements (such as italics for the title). For most member sites it holds that even if HTML elements are used usually authors are merely separated by commas.

Furthermore, the table indicates whether additional information to author, title, and year are available and how complete the information is (if e.g. year or conference is missing). The least common denotator for all member pages are the availability of author names, title name and publication year, in some cases additionally abstracts and links are available.

4.2 Gathering Web Data for the Personal Publication Reader

In the following, we describe a step-by-step construction of this example from the viewpoint of an application designer who creates this application.

A human being tends to assign semantic meaning to parts of a Web page; a designer does not think of *table row* as of a set with text values, but rather as a *publication entry*. Therefore, the basic building block of a wrapper program is a so-called *pattern*, a container for pieces of information with the same meaning. Patterns are structured in a hierarchical fashion. In the lower half of the Visual Wrapper's UI (see Figure 2) an active example Web page is displayed for marking example instances: For each type of Web page, an own wrapper has to be created; in the following the wrapper creation for the publications of Munich is illustrated.

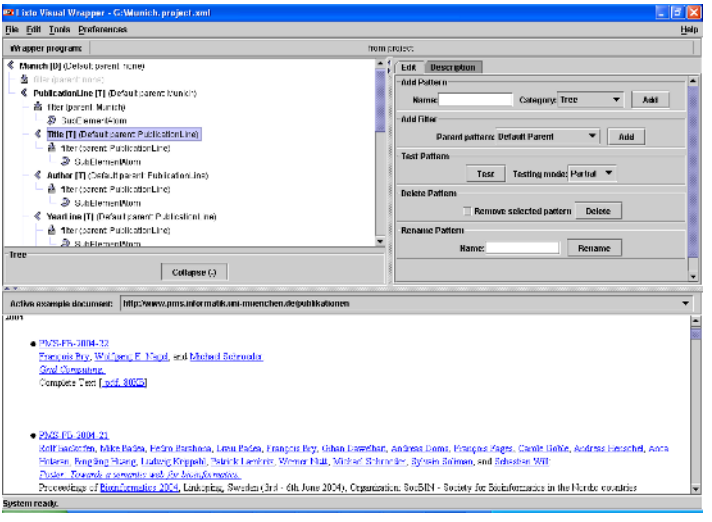

Fig. 2. Lixto Visual Wrapper: Wrapping Publication Pages

In this case, the designer identifies one of the list items (each resembling a publication) as a pattern *PublicationLine*. Once a pattern is created, the designer continues with visually defining a filter, a crucial part of the pattern which defines how to extract relevant information from its parent pattern instances. Internally, filters are represented in Elog, but the language is entirely hidden from the wrapper designer.

Defining a filter expects the designer to select an example publication with two mouse clicks on the example Web page. A filter definition continues with optional fine-tuning of properties for the generated generalization of the chosen example. It is possible to visually debug the wrapper program, i.e., to test filters. Typically, operators test filters after adding new components. Based on results, the designer decides whether to extend (i.e., add a filter) or shrink (i.e., add condition to an existing filter) the set of matched instances.

In this example, the system displays the complete list of matched publications for the so-far created filter by highlighting parts of the Web page. In cases where the system generalization does not detect all instances correctly, additional conditions can be imposed.

Next a child pattern *Title* of the just defined pattern is created and then a filter with the condition that the extracted element is in italics. The pattern *Author* on the Munich page can be easily characterized, too, by the fact that a special hyperlink is present and that the author names precede the title.

On other pages such as e.g. Linköpping the extraction of authors is more advanced. Some authors are inside hyperlinks, others merely separated by commas. Moreover, on other sources authors are sometimes incorrectly splitted, names abbreviated and different separators used. Therefore, we developed an author concept based on all detected variations.

On the Munich page the year can be extracted from several places (see Figure 2). One possibility is from the internal number. The first line of the list item is extracted, and in a subsequent step the four digit number is taken out. On some other sources the year has to be extracted from the headline, and in a subsequent step mapped to each entry.

In a similar fashion the remaining patterns are defined and the wrapper is stored. The XML Companion of the publication Web page that can be regularly generated by applying the wrapper is comprised of entries like the one given below:

```
<Publication>
  <Title>Visual Exploration and Retrieval of XML Document
        Collections with the Generic System X2</Title>
  <Author>Holger Meuss</Author>
    [...more authors...]
  <Year>2004</Year>
  <Link>http://www.pms.informatik.uni-muenchen.de/
        publikationen/PMS-FB/PMS-FB-2004-12.pdf</Link>
</Publication>
```

As next step the XML data of the various sources has to be combined, cleaned, syndicated into the ontology, and regularly scheduled. These operations are carried out by configuring a visual information flow in the *Lixto Transformation Server* as described in Section 4.3.

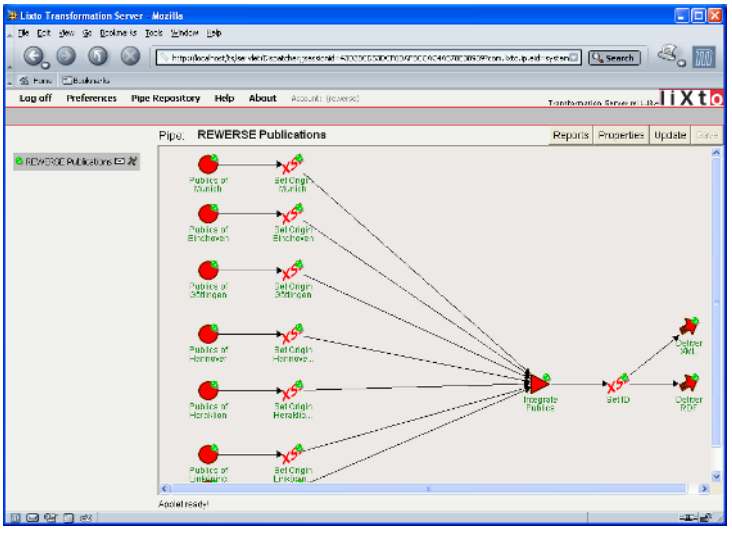

Fig. 3. Lixto Transformation Server: REWERSE Publication Data Flow

4.3 Visual Data Aggregation for the Personal Publication Reader

In the Personal Publication Reader scenario, the application designer visually composes the information flow from Web sources using the Lixto Transformation Server to an RDF presentation that is handed over to the Personal Publication Reader once a week.

First, the application designer creates Source components that contain Lixto wrappers. In the source components (that are reflected as disks in Figure 3) a schedule is defined how often which Web source is queried and Deep Web navigation sequences containing logins and forms can be stored. Next, the wrapper designer can combine the XML documents by adding integration components.

In the "XSL" components publication data is harmonized to fit into a common structure, an attribute "origin" is added containing the institution's name, and author names are harmonized by being mapped to a list of names known by the system. The triangle in Figure 3 represents a data integration unit; here data from the various institutions is put together and duplicate entries are removed. IDs are assigned to each publication in the subsequent step. Finally, the XML data structure is mapped to a defined RDF structure (this happens in the lower arc symbol in Figure 3) and passed on to the Personal Publication Reader as described below. A second deliverer component delivers the XML publication data additionally. One sample RDF output entry is depicted below:

```
<rdf:Description
   rdf:about="http://www.pms.informatik.uni-muenchen.de/
   publikationen/PMS-FB/PMS-FB-2004-12.pdf">
  <dc:publisher>University of Munich</dc:publisher>
  <dc:title>Visual Exploration and Retrieval of XML Document
   Collections with the Generic System X2</dc:title>
  <dc:creator>
    <rdf:Seq>
        <rdf:li rdf:resource="#Holger Meuss"/>
        <rdf:li rdf:resource="#Klaus U. Schulz"/>
        <rdf:li rdf:resource="#Felix Weigel"/>
        <rdf:li rdf:resource="#Simone Leonardi"/>
        <rdf:li rdf:resource="#Francois Bry"/>
    </rdf:Seq>
  </dc:creator>
  <dc:date>2004</dc:date>
  <dc:identifier>http://www.pms.informatik.uni-muenchen.de/
   publikationen/PMS-FB/PMS-FB-2004-12.pdf</dc:identifier>
</rdf:Description>
```

This application can be easily enhanced by connecting further Web sources. For instance, abstracts from `www.researchindex.com` can be queried for each publication lacking this information and joined to each entry, too. Moreover, using text categorization tools one can rate and classify the contents of the abstracts. Another possibility is to extract organization and people data from the institution's Web pages to inform the ontology to which class in the taxonomy an author belongs (such as full professor).

4.4 Modeling Domain Knowledge: The REWERSE Ontology

In addition to the extracted information on research papers that we obtain as described in the previous section, we collect the data about the members of the research project from the member's corner of the REWERSE project. We have constructed an ontology for describing researchers and their involvement in scientific projects like REWERSE. This "REWERSE-Ontology" has been built using the Protégé tool [16]. It extends the Semantic Web Research Community Ontology (SWRC) [19]. An excerpt of the REWERSE-Ontology, written in OWL[14]:

```
<owl:ObjectProperty rdf:ID="hasStaffMember">
 <rdfs:subPropertyOf>
   <owl:ObjectProperty rdf:about="#hasMember"/>
 </rdfs:subPropertyOf>
 <owl:inverseOf>
   <owl:ObjectProperty rdf:ID="employedAt"/>
 </owl:inverseOf>
 <rdfs:label xml:lang="de">Angestellte</rdfs:label>
 <rdfs:domain>
   <owl:Class>
     <owl:unionOf rdf:parseType="Collection">
       <owl:Class rdf:about="#University"/>
       <owl:Class rdf:about="#Institute"/>
       <owl:Class rdf:about="#Project"/>
       <owl:Class rdf:about="#Department"/>
       <owl:Class rdf:about="#Company"/>
     </owl:unionOf>
   </owl:Class>
 </rdfs:domain>
 <rdfs:range rdf:resource="#Person"/>
 <rdfs:label xml:lang="en">Staffmember</rdfs:label>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="#employedAt">
 <rdfs:label xml:lang="en">employed at</rdfs:label>
 <rdfs:range>
   <owl:Class>
     <owl:unionOf rdf:parseType="Collection">
       <owl:Class rdf:about="#Project"/>
       <owl:Class rdf:about="#Institute"/>
       <owl:Class rdf:about="#University"/>
       <owl:Class rdf:about="#Department"/>
       <owl:Class rdf:about="#Company"/>
     </owl:unionOf>
   </owl:Class>
 </rdfs:range>
 <rdfs:subPropertyOf rdf:resource="#involvedIn"/>
 <rdfs:label xml:lang="de">angestellt bei</rdfs:label>
 <rdfs:domain rdf:resource="#Person"/>
 <owl:inverseOf rdf:resource="#hasStaffMember"/>
</owl:ObjectProperty>
```

To match the domain knowledge in the REWERSE Researcher Ontology to the extracted publication data, we have a resource identification problem. The author names may vary - for example, `F. Bry, Franqis Bry, Prof. F. Bry, etc.` . A "helper" ontology, describing the full name of each author, and a variety of commonly used designators of his or her name, is currently used to solve this matching task.

4.5 Content Syndication and Personalized Views

As we have described in the previous sections, we have extracted relevant data from various, non-uniform Web sites, and created an extension of the SWRC ontology to model the needs of scientific projects such as REWERSE. We will now see how personalization rules reason about this collected data in order to syndicated and personalize the view on the data. A discussion on personalization reasoning for the Semantic Web can be found in [1]. As an example, the following rule (using the TRIPLE[18] syntax) determines all authors of a publication:

```
FORALL A, P all_authors(A, P) <-
  EXISTS X, R (
  P['http://.../rewerse#':author -> X]@'http:...#':publications
  AND X[R -> 'http://www.../author':A]@'http:...#':publications).
```

Further rules combine information on these authors from the researcher ontology with the author information. E.g. the following rule determines the employer of a project member, which might be a company, or a university, or, more generally, some instance of a subclass of an organization:

```
FORALL A,I works_at(A, I) <-
   EXISTS A_id,X (name(A_id,A)
     AND ont:A_id[ont:involvedIn -> ont:I]@'http:...#':researcher
     AND ont:X[rdfs:subClassOf ->
               ont:Organization]@rdfschema('http:...#':researcher)
     AND ont:I[rdf:type -> ont:X]@'http:...#':researcher).
```

For a user with specific interests, for example "interest in personalized information systems", information on respective research groups in the project, on persons working in this field, on their publications, etc., is syndicated. As an example, the following rule derives all persons working in specific working groups in the project. Personalization is realized by matching the results of this rule with the individual request, e.g `ont:WG[ont:name -> 'WG A3 - Personalized Information Systems')`.

```
FORALL WG,M working_group_members(WG,M) <-
       ont:WG[rdf:type -> ont:WorkingGroup]@'http:..#':researcher
       AND ont:WG[ont:hasMember-> ont:M]@'http://...#':researcher.
```

For the PPR, we instantiated a personalization service in the Personal Reader framework which holds the above mentioned rules, and further personalization rules of the PPR. An appropriate visualization service for creating the user interface has been implemented. The screenshot in Figure 4 depicts the output of the visualization service of the PPR.

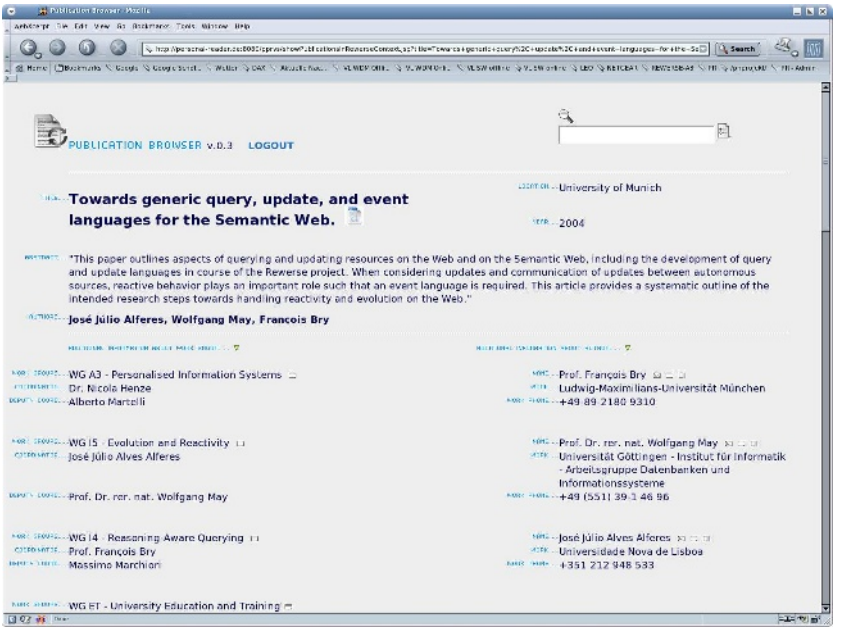

Fig. 4. Screenshot of the Personal Publication Reader, showing the syndicated view on publications in REWERSE, the context in the project in which this research has been done, together with the appropriate links, and additional information about the authors of the publication like homepage, phone number, etc. The Personal Publication Reader is available via the URL www.personal-reader.de

5 Conclusion

This paper describes an approach for realizing advanced personalized information systems in the Semantic Web. We discuss our approach by means of an example application, a Personal Publication Reader, which provides a personalized, syndicated view on distributed, non-uniform web data. The information provision part for the Personal Publication Reader is solved by using the Lixto approach. *Lixto* is an easily accessible technology based on a solid theoretical framework [2, 3, 7] and a visual approach that allows application designers to define continuously running information agents fetching data from the Web. Many functions that will be tangible only in the future "Semantic Web" can be crucially supported by the usage *Lixto*. Content syndication and personalization is achieved by reasoning about ontological knowledge and extracted Web data. The Personal Publication Reader is realized using the Personal Reader Framework for designing, implementing, and maintaining personalized Web Content Readers. Until know, we have realized such Readers for e-Learning and for publication browsing, ongoing work focuses on implementing additional personalization services, and on on improving the service orchestration functionality in our frame-

work. In future work we will continue our approach of realizing *Personalization Services for the Semantic Web*.

References

1. G. Antoniou, M. Baldoni, C. Baroglio, R. Baumgartner, F. Bry, T. Eiter, N. Henze, M. Herzog, W. May, V. Patti, S. Schaffert, R. Schindlauer, and H. Tompits. Reasoning methods for personalization on the semantic web. *Annals of Mathematics, Computing & Telefinformatics*, 2(1):1–24, 2004.
2. R. Baumgartner, S. Flesca, and G. Gottlob. Visual web information extraction with Lixto. In *Proc. of VLDB*, 2001.
3. R. Baumgartner, M. Herzog, and G. Gottlob. Visual programming of web data aggregation applications. In *Proc. of IIWeb-03*, 2003.
4. T. Berners-Lee. The semantic web - mit/lcs seminar, 2002. http://www.w3c.org/2002/Talks/09-lcs-sweb-tbl/.
5. T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, May 2001.
6. S. Flesca, G. Manco, E. Masciari, E. Rende, and A. Tagarelli. Web wrapper induction: a brief survey. *AI Communications Vol.17/2*, 2004.
7. G. Gottlob and C. Koch. Monadic datalog and the expressive power of languages for Web Information Extraction. In *Proc. of PODS*, 2002.
8. N. Henze and M. Herrlich. The Personal Reader: A Framework for Enabling Personalization Services on the Semantic Web. In *Proceedings of the Twelfth GI-Workshop on Adaptation and User Modeling in Interactive Systems (ABIS 04)*, Berlin, Germany, 2004.
9. N. Henze and M. Kriesell. Personalization functionality for the semantic web: Architectural outline and first sample implementation. In *Proccedings of the 1st International Workshop on Engineering the Adaptive Web (EAW 2004), co-located with AH 2004*, Eindhoven, The Netherlands, 2004.
10. N. Henze and W. Nejdl. A logical characterization of adaptive educational hypermedia. *New Review of Hypermedia*, 10(1), 2004.
11. SUN - java Server Pages, 2004. http://java.sun.com/products/jsp/.
12. S. Kuhlins and R. Tredwell. Toolkits for generating wrappers. In *Net.ObjectDays*, 2002.
13. A. H. Laender, B. A. Ribeiro-Neto, A. S. da Silva, and J. S. Teixeira. A brief survey of web data extraction tools. In *Sigmod Record 31/2*, 2002.
14. OWL, Web Ontology Language, W3C Recommendation, Feb. 2004. http://www.w3.org/TR/owl-ref/.
15. OWL-S: Web Ontology Language for Services, W3C Submission, Nov. 2004. http://www.org/Submission/2004/07/.
16. Protege Ontology Editor and Knowledge Acquisition System, 2004. `http://protege.stanford.edu/`.
17. RDF Vocabulary Description Language 1.0: RDF S, 2004. `http://www.w3.org/TR/2004/REC-rdf-schema-20040210/`.
18. M. Sintek and S. Decker. TRIPLE - an RDF Query, Inference, and Transformation Language. In I. Horrocks and J. Hendler, editors, *International Semantic Web Conference (ISWC)*, pages 364–378, Sardinia, Italy, 2002. LNCS 2342.

19. SWRC - Semantic Web Research Community Ontology, 2001. http://ontobroker.semanticweb.org/ontos/swrc.html.
20. WSDL: Web Services Description Language, version 2.0, Aug. 2004. http://www.w3.org/TR/2004/WD-wsdl20-20040803/.
21. XML: extensible Markup Language, 2003. http://www.w3.org/XML/.
22. XML-based RPC: Remote procedure calls based on xml, 2004. http://java.sun.com/xml/jaxrpc/index.jsp.