Organising Web links semantically with Boek

[1]Rakesh Paul, [2]Paul Mc Kevitt, [3]John Macrae

[1, 2] School of Computing & Intelligent Systems, University of Ulster, Magee,
Derry/Londonderry BT48 J7L, Northern Ireland
[3] Office of Innovation, University of Ulster, Jordanstown,
Newtownabbey BT37 0QB, Northern Ireland
Paul-r2@email.ulster.ac.uk

Abstract. The World Wide Web serves as the primary source of many information requirements. The Semantic Web and related technologies provide an infrastructure to organise data based on its semantics. Since the web has grown exponentially, it is difficult to keep track of all individual preferences. The system proposed here, Boek, enables users to organise semantically a subset of web data. Boek can store links to web pages and retrieve them later based on semantic information. Boek also holds reserved space for its users to store their favourite data semantically. The individual data models can then be combined to form a universal data store which can then act as a knowledge base for everyone. Future work will involve extending Boek to address a number of domains such as business, medicine, education and the creative industries.

Keywords. Boek, Semantic Web, Ontologies, Knowledge Management, User Interface, Metadata, Ontology learning, RDF, Java, Jena

1 Introduction

The World Wide Web is an invaluable resource for information search and web browsers act as the platform for access. However, storing the information collected from various sites according to user preferences and meaning is not always given a priority. Users must depend on search engines to locate web pages of their interest. As an aside, one might consider whether users are reliably given accurate information when they 'search' the Internet using a particular search engine. The user may be forced to repeat the searching process again and again to retrieve the same data over a period. Bookmarks within web browsers enable users to store links they prefer. However, normally bookmarks don't provide the ability to group related information based on its contents. Bookmarks are stored locally confining their global use. Then there are websites like Digg and Delicious which enable bookmark synchronisation with search and web search based on bookmarks created by users. However, these sites usually don't provide information organisation based on semantics.

This paper discusses Boek (German for 'book'), an online bookmark storage system developed on Semantic Web [1] technology. Boek enables users to enhance their

information base with favourite links and lookup for links based on titles, subjects, authors and URLs. Boek utilises some of the latest graphical user design techniques and open source tools for accessing Semantic Web capabilities. Boek's open source framework implementation can be used to develop any semantic web based applications. This paper explores requirements analysis, architectural design, implementation and evaluation of Boek. The paper is organised as follows: Section 2 discusses background and literature review; Section 3 discusses Design and implementation of Boek; Section 4 discusses testing and evaluation of Boek and Section 5 relates Boek to other work. Finally, Section 6 discusses conclusions and future work that can extend Boek's architecture, not just for bookmarks, but to any useful information content.

2 Background and Literature Review

In this section we discuss semantic web fundamentals and related work on applications for generating and storing data based on its semantics.

2.1 Semantic Web Fundamentals

The Semantic Web enables integration of diverse knowledge bases to a single point of access. The following features of the Semantic Web can be utilised to organise and group data.

2.1.1 Explicit metadata

Currently the web holds data that is more useful to humans than to computers. The Semantic Web tackles this issue by introducing metadata [2], which is data about data. In addition to containing formatting information aimed at humans, metadata can also contain information about its content. Thus metadata captures part of the meaning of data. The Semantic Web employs languages like RDF (Resource Description Framework) [2] to define the syntax of metadata information. For example, the RDF data for a health treatment centre is shown in Figure 1.

2.1.2 Ontologies

Ontologies [3] are vocabularies for expressing metadata. Typically, ontologies consist of a finite list of terms, and relationships between these terms. The terms denote important concepts (classes of objects) of the domain. For example, in a university setting, staff members, students, courses, lecture rooms and disciplines are some important concepts. Relationships define how these concepts are interrelated to give a meaning to the university setting. Ontologies provide shared understanding for a domain.

The ontology vocabularies are defined using RDFS (Resource Description Framework Schema) and OWL (Web Ontology Language) [4] description languages. RDF Schema serves as the meta-language or vocabulary to define properties and classes of RDF resources. OWL has a richer vocabulary which describes properties and classes, relations between classes (e.g. disjointness), cardinality (e.g., "exactly one"), equality, and characteristics of properties (e.g., symmetry).

```
<company>
     <treatmentOffered>Ayurvedic Rejuvenation </treatmentOffered>
     <companyName>Heritage of India</companyName>
     <location>
          <country>India</country>
          <state>Kerala</state>
          <city>Palagath</city>
     </location>
</company>
```

Fig. 1. Encoding metadata information for a health treatment center

2.1.3 Logic and Proof

Logic [2] derives conclusions from a set of relations. Logic facilitates inference about knowledge and derives explicit data from implicit data. In the Semantic Web context logical reasoning establishes consistency and correctness of data sets. For instance, consider the following example where the Semantic Web can infer some explicit data from the available knowledge:

Flipper is a Dolphin (known relationship)
Every Dolphin is also a Mammal (known relationship)
Flipper is a Mammal (inferred relationship using logic)

The advantage of Logic is that it can explain how conclusions are derived. That is, logic can retrace the entire procedure involved in deriving a conclusion. *Proofs* trace or explain the steps of logical reasoning.

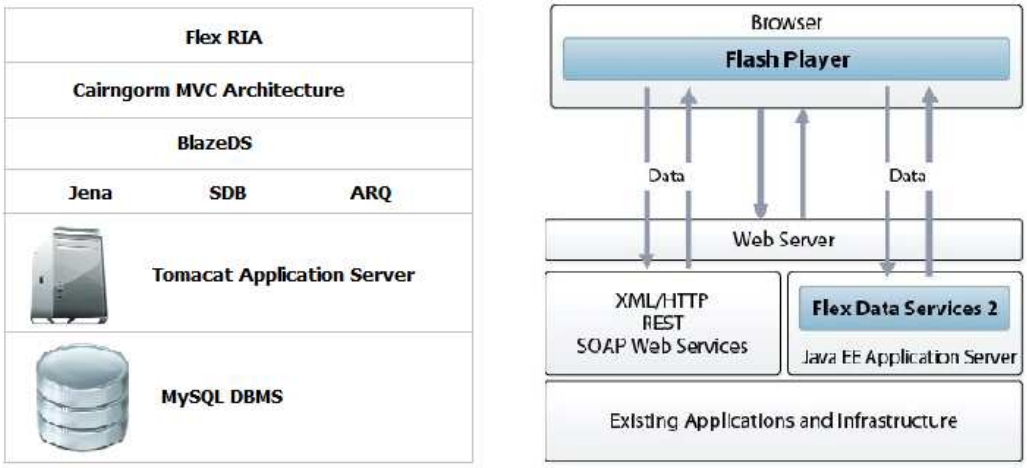
2.2 Semantic Web Applications

There are solutions like MIT lab's Piggy Bank [5] which can facilitate generation and storage of semantic data from existing web pages. Piggy Bank extracts information from existing Web pages and stores it in RDF. Piggy Bank employs custom software for screen scraping and storing extracted data locally. There is also an option for users to publish this data to a Semantic Bank [5] (Server for storing semantic data) and use it later on demand. The Semantic Bank can then be used by other users to find more granular information on different topics. SemCards [6] are machine and human-readable entities that allow non-experts to create and use semantic content with ease, whilst allowing machines to participate in the knowledge organisation process.

Wilks [7] discusses three distinct views on the Semantic Web, principally with reference to the issue of the relationship of natural language structure to knowledge representation and asks the question, "what are its semantics?". Mc Kevitt [8] discusses the requirement for semantic representations to be multimodal.

3 Boek Design and Implementation

Our system, Boek, is a Rich Internet Application (RIA) aimed at providing an engaging experience to its users. Boek integrates seamlessly the most recent advancements in RIA and Semantic Web technologies with the ever popular Java and MySQL [9] development environments. Figure 2 shows the architectures of Boek and Flex [10]. Boek's user interface consists of Adobe Flex designed web pages. The middle tier uses Cairngorm Model View Controller (MVC) [11] architecture and BlazeDS [12] for remote messaging services. Jena API [13] forms the semantic web based database tier. The semantically organised data is persisted to a MySQL database.



(a) (b)

Fig. 2. Boek (a) and Flex (b) architectures

3.1 Software Analysis

Software utilised in Boek includes Adobe Flex, Cairngorm, BlazeDS, RDF, Jena Semantic Web Framework, SPARQL with Jena SDB and ARQ APIs.

Adobe® Flex® [10] is a Rich Internet application (RIA) offering a rich, engaging experience that improves user satisfaction and increases productivity. It is an open source framework for developing intuitive web applications by leveraging the Adobe® Flash® Player and Adobe AIR® runtimes. RIA combines the best of desktop, web and communication technologies to provide better quality web applications.

Flex produces Flash player compatible files which can run in a flash player enabled browser. Representational State Transfer (REST) is a style of software architecture for distributed hypermedia systems such as the World Wide Web.

Cairngorm [11] is a framework used to deliver successful Flex projects in enterprise. Cairngorm is an ActionScript library for building applications based on the Model-View-Controller (MVC) design pattern. It is an approach for organising and partitioning code and packages, component functionality and roles. Cairngorm encourages developers to identify, organise, and separate code based on its roles/responsibilities. The Flex and Cairngorm micro architecture forms the client side logic of Boek.

BlazeDS [12] is a free open source server-based Java remoting and web messaging technology that enables developers to easily connect to back-end distributed data and push data in real-time to Adobe® Flex and Adobe AIR™ applications for more responsive RIA experiences. BlazeDS provides a set of services that helps a client-side application to connect to server-side data, and pass data amongst multiple clients connected to the server. BlazeDS implements real-time messaging between clients.

A BlazeDS application consists of two parts: (1) The client-side application: A BlazeDS client application is typically an Adobe Flex or AIR application. Flex and AIR applications use Flex components to communicate with the BlazeD Server; (2) The BlazeDS server: The BlazeDS server runs in a web application on a J2EE application server. BlazeDS includes three preconfigured web applications that can be used as the basis of application development. Boek utilises the messaging application of the BlazeDS.

Resource Description Framework (RDF) [2] plays the role of describing data and meta-data. RDF is written in XML and is designed to be read and understood by computers and is not designed for being displayed to end users. The fundamental concepts of RDF are: (1) *Resources* is an object; a "thing" that is being defined. Resources may be e.g. authors, books and publishers; (2) *Properties* are special kinds of resources, and describe relations between resources, e.g. "written by", "age" and "title"; (3) *Property Values* can either be resources, or literals [atomic values (strings)]. Figure 3 shows different RDF concepts.

RDF identifies concepts using Web identifiers (URIs), and describes resources with properties and property values.
Statements assert the properties of a resource. A statement is an object-attribute-value triple, consisting of: (1) Resource (2) Property (3) Property Value.
Consider the statement, "The author of http://www.something.com/rdf is someone". The subject of the above statement is: http://www.something.com/rdf, the predicate is: author and the object is: Someone. Figure 4 shows the graphical representation of the above mentioned RDF statement.

Fig. 3. RDF resources, property and property values

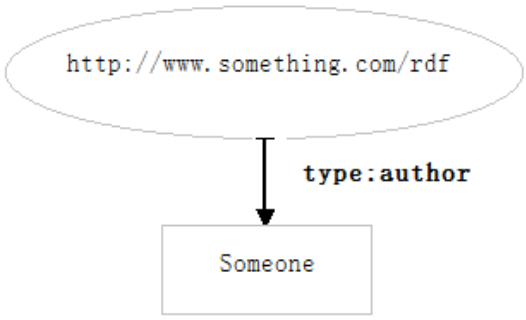


Fig. 4. RDF graph representation of a statement

Jena Semantic Web Framework [13] is a Java framework for building Semantic Web applications. It provides a programming environment for RDF, RDFS (Resource Description Framework Schema), OWL (Web Ontology Language), and SPARQL (Query Language for RDF) and includes a rule-based inference engine. Jena can facilitate creating and manipulating RDF graphs like that shown in Figure 4. Jena has object classes to represent graphs, resources, properties and literals. The interfaces representing resources, properties and literals are called Resource, Property and Literal respectively. In Jena, a graph is called a model and is represented by the Model interface. The code to create the graph, or model, is shown in Figure 5.

```
SELECT ?x ?author
WHERE {
      ?x type:author ?author
}

Executing the query will give the following result
```

x	author
http://www.something.com/rdf is someone"	Someone

Fig. 5. A sample SPARQL query for the RDF graph shown in Fig. 4

SPARQL [14] is a query language and data access protocol for the Semantic Web. Data access means reading information, not writing (updates). SPARQL is defined in terms of a RDF data model and will work for any data source that can be mapped into RDF. SPARQL can be used to retrieve information from a RDF graph. SPARQL can also construct information dynamically from the existing graphs.

SDB [15] is a Jena component for storage and querying, specifically for the SPARQL language. The RDF model is created in a database rather than in files enabling massive RDF data storage. Most of the popular databases are supported by SDB. Boek uses MySQL as its database management system for storing the RDF data. The general syntax for creating a model with SDB is shown below in Figure 6. Though SDB provides an inbuilt querying API, Boek utilises the more powerful Jena ARQ API for SPARQL querying.

```
// Create database connection to the desired database
Connection connection = DriverManager.getConnection("database url");

// Create a SDB database connection using the above connection
DBConnection dbConnection = new DBConnection(connection, 'Mysql')
// Mysql being the dbms type

//Create RDF model in the db
Model model = ModelRDB.createModel(dbConnection, "modelname");
```

Fig. 6. Code for creating RDF model using SDB

ARQ [16] is a query engine for Jena that supports the SPARQL RDF Query language. ARQ provides java classes and interfaces that will facilitate constructing and executing queries. ARQ also provides functionalities to parse the results of query execution against a RDF data store. Sample code for querying Boek's knowledge base is shown below in Figure 7.

MySQL [9] is a relational database management system (RDBMS). MySQL provides a resource for consistent, reliable and scalable RDF storage for Boek.

3.2 Boek Implemenation

Implementation of Boek includes the front end (graphical user interface) design using Adobe Flex and middle tier design using the Adobe Cairngorm framework. Boek utilises the BlazeDS messaging service to interact with its backend database. Once a request for semantic lookup reaches the backend server, Jena and ARQ API's interact with semantic web data stored using SDB to retrieve necessary information. The results are then parsed and send back to the front end.

Appendixes A shows screen shorts of the Boek interface. Figure 8 shows the 'Find/Search' user interface of Boek. Users can search for bookmarks based on dif-

ferent properties of a stored bookmark such as title, author, URL and description. They can also search using multiple keywords as shown in Figure 9. Figure 10 shows the result of adding a new bookmark to Boek. Figure 13 shows the SPARQL query with multiple search values for obtaining the results shown in Figure 10.

```
// Sparql query string
String queryString = "SELECT ?x ?author" WHERE {?x type:author ?author}"

// Creating a Query Object from the query string
Query query = QueryFactory.create(queryString);

// Creating Query execution for
QueryExecution qexec = QueryExecutionFactory.create(query, model1);
// model1 being an RDF graph

// Executing the query against the model1
rs = qexec.execSelect() ;

// Iterate the result of query execution
while (rs.hasNext()) {

// Get the query solution for parsing
      QuerySolution qs = rs.nextSolution();

      // parse the query solution for individual result
      parse(qs);
}
```

Fig. 7. Jena ARQ code for querying SPARQL

4 Testing and Evaluation of Boek

The testing process applied to Boek falls under the following categories: (1) Developer Testing; (2) Integration Testing; (3) Graphical User Interface (GUI) Testing and (4) User Acceptance Testing (UAT).

Developer testing was performed at different development phases to ensure Boek satisfied the requirements specified at the beginning of that phase. Each module developed was tested to check it was performing according to the requirements. Both the user interface and backend Java code were subjected to this process. Java modules were tested by creating standalone applications and passing test data to the methods from the standalone applications. Once the method passed this test, it was then integrated within Boek. Integration testing helped to identify problems that occurred when independently developed modules were added to Boek. This was particularly helpful to spot errors when standalone Java applications were integrated to Boek.

Graphical User Interface (GUI) testing was performed to ensure the accessibility, responsiveness and efficiency of Boek. Accessibility testing checked whether users can enter, navigate and exit from the different views of Boek with ease. Responsiveness of Boek analysed whether users were receiving appropriate response messages for their actions. Finally, efficiency of Boek defined the pace of responsiveness. Since Boek utilises RIA capabilities this testing was critical. User Acceptance testing was performed to analyse the usefulness of Boek. The suggestions received were then updated within the initial requirements for Boek.

Boek was primarily designed to evaluate the capabilities of the semantic web to store data semantically. Boek enables users to search bookmarks based on title, author, description or URL. The information stored using Boek is limited to bookmarks. However, utilising full capabilities of the semantic web requires the creation of ontologies [17]. Use of more powerful languages like RDFS and OWL is necessary to provide information based on logic and inference. Ontology learning [18] is essential for creating dynamic meaningful ontologies from existing data stored as RDF.

5 Relation to other work

Boek's architecture is highly scalable by providing a clear isolation between the user interface and the semantic web based backend. Boek was designed taking inspiration from MIT PiggyBank [5] and SemCards [6] with a view to scale them to a personal knowledge repository. Like in PiggyBank, a bookmark once stored can be retrieved by any other user based on semantics. Users can also recommend links to friends or colleagues. What distinguishes Boek from other social bookmarking web sites like Digg and Delicious is its ability to store information about personal documents with slight modification.

6 Conclusion and future work

The World Wide Web facilitates universal collaboration of information spaces. The openness of the web has come with a price. Web pages holding valuable information are almost hidden from most expected users due to information overload. Semantic Web technologies were proposed as a solution to add meaning to information on the web. However, applying the Semantic Web to billions of existing web pages is challenging. This can be achieved by enabling individual web users to create their own semantic web content. The system Boek discussed here utilises capabilities of Semantic Web technologies to provide a meaningful bookmarking technique. Boek is driven by open source technologies such as Adobe Flex, Jena API and MySQL DBMS resulting in a cost effective application with high scalability.

Future work on Boek includes replacing the Flex based GUI with a web browser plug-in. Currently Boek only enables the addition of bookmarks based on semantics.

However, with little effort Boek can be enhanced to organise all individual information requirements. Ontology learning extracts relevant domain terms from a different link content and relates them to appropriate concepts in a general-purpose ontology [19], [20], [21]. This will eliminate the requirement for users to add semantic information about links. Furthermore, this can enable building clusters of domain data which can be shared between different users. Also, facilities can be added to automatically publish useful links and information from one user to others.

7 References

1. Berners-Lee, T., Hendler, J. and Lassila, O.: The semantic web - a new form of web content that is meaningful to computers will unleash a revolution of new possibilities. In Scientific American 284. (2001) 34–43, May
2. Passin, T. B.: Explorer's Guide to the Semantic Web. Greenwich, CT: Manning (2004)
3. van Harmelen, F.: Semantic Web Technologies as the Foundation of the Information Infrastructure. In: P. van Oosterom, S. Zlatanove (eds.), Creating Spatial Information Infrastructures: Towards the Spatial Semantic Web, 37-54. London: CRC Press (2008)
4. Janev, V., Vranes, S.: Semantic Web Technologies: Ready for Adoption?, IT Professional, Vol. 11, No. 5. (2009) 8-16
5. Huynh, D., Mazzocchi, S., Karger, D.: Piggy Bank: Experience the Semantic Web Inside Your Web Browser, Lecture Notes in Computer Science, Vol. 3729. (2005) 413-430
6. Thórisson, K.R., Spivack, N., Wissner, J.M.: SemCards: A New Representation for Realizing the Semantic Web. ICCCI-2009, Wroclaw, Poland, October 5-7. (2009) 425-436
7. Wilks, Y.: The Semantic Web: Apotheosis of Annotation, but What Are Its Semantics?, Intelligent Systems, IEEE , Vol.23, No.3. (2008) 41-49, May-June
8. Mc Kevitt, P.: Advances in Intelligent MultiMedia: MultiModal semantic representation, In Proceedings of the Pacific Rim International Conference on Computational Linguistics (PACLING-05), Hiroshi Sakaki (Ed.), Meisei University (Hino Campus), Hino-shi, Tokyo Japan. (2005) 2-13, August
9. MySQL: http://www.mysql.com/ (2011)
10. Adobe Flex: http://www.adobe.com/products/flex/ (2011)
11. Adobe Flex Cairngorm: http://opensource.adobe.com/wiki/display/cairngorm/Cairngorm (2011)
12. Adobe Flex BlazeDS: http://opensource.adobe.com/wiki/display/blazeds/BlazeDS (2011)
13. Jena API: http://jena.sourceforge.net/ (2011)
14. SPARQL Query Language for RDF: http://www.w3.org/TR/rdf-sparql-query/ (2011)
15. Jena SDB database for Jena: http://openjena.org/SDB/ (2011)
16. Jena ARQ API: http://jena.sourceforge.net/ARQ/app_api.html (2011)
17. Hepp, M.: Possible Ontologies: How Reality Constrains the Development of Relevant Ontologies, , IEEE Internet Computing, Vol. 11, No. 1. (2007) 90-96
18. Maedche, A.; Staab, S.: Ontology learning for the Semantic Web, IEEE Intelligent Systems , Vol.16, No. 2. (2001) 72- 79, Mar-Apr
19. Alani, H., Hall, W., O'Hara, K., Shadbolt, N., Szomszor, M. & Chandler, P,: Building a Pragmatic Semantic Web, IEEE Intelligent Systems, Vol. 23, No. 3. (2008) 61-68
20. Benjamins, V.R.: Near-Term Prospects for Semantic Technologies, IEEE Intelligent Systems, Vol. 23, No. 1. (2008) 76-88

21. d'Aquin, M., Motta, E., Sabou, M., Angeletou, S., Gridinoc, L., Lopez, V. & Guidi, D.: Toward a New Generation of Semantic Web Applications, IEEE Intelligent Systems, Vol. 23, No. 3. (2008) 20-28

Appendix A

Fig. 8. Search title

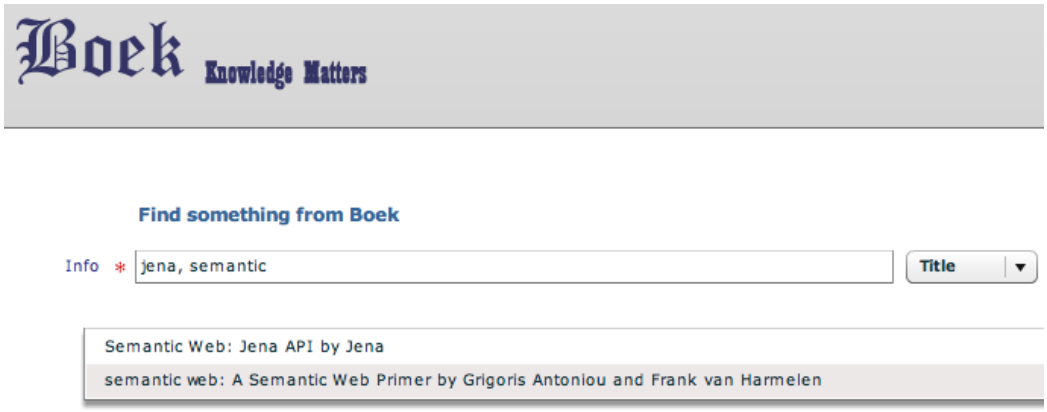


Fig. 9. Search using multiple terms

Fig. 10. Add data result showing matching topic on subject added

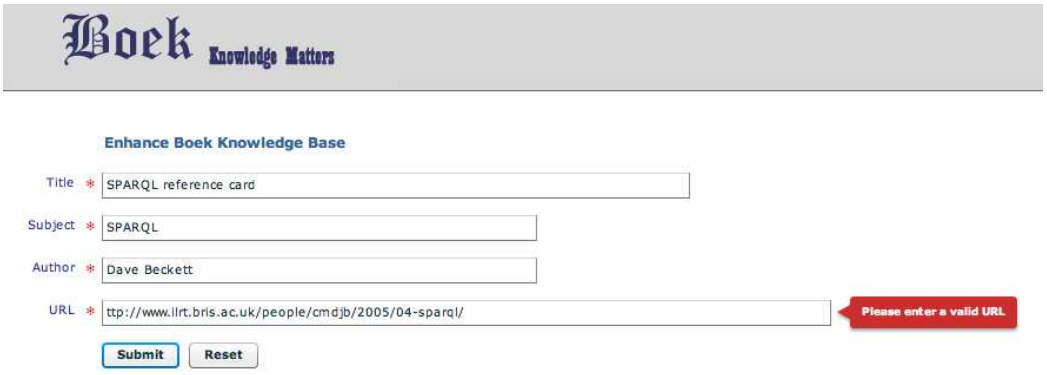


Fig. 11. Invalid URL validation error

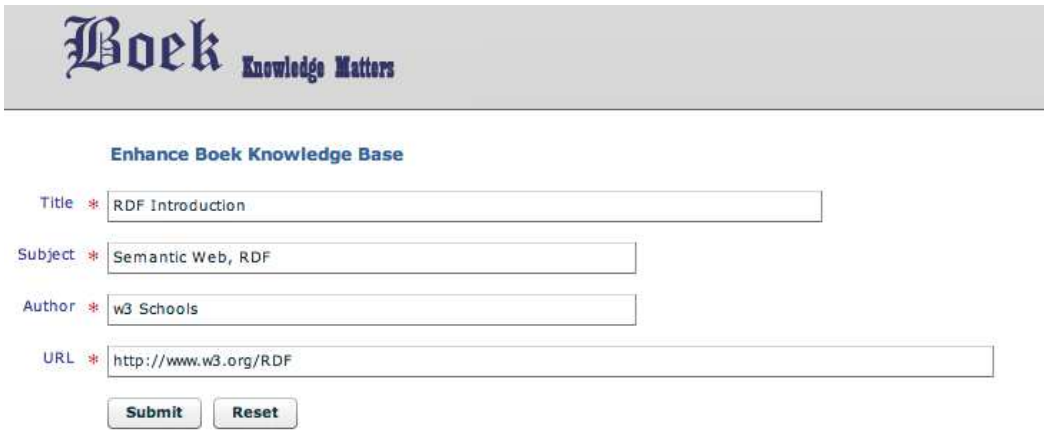


Fig. 12. Add data form without any errors

```
PREFIX  boek: <http://www.traume.com/elements/1.0/>
SELECT  ?author ?subject ?title ?topicOf
WHERE
  { ?x  boek:author   ?author ;
        boek:subject  ?subject ;
        boek:title    ?title ;
        boek:topicOf  ?topicOf .
    FILTER ( regex(?title, "jena", "i") || regex(?title,
"semantic", "i") )
  }
```

Fig. 13. SPARQL query with multiple search values