A Data-Model Driven Web Application Development Framework

CONFERENCE PAPER · JANUARY 2014

3 AUTHORS:

Erdogan Dogdu

TOBB University of Economics and Technology

52 PUBLICATIONS **291** CITATIONS

SEE PROFILE

Sherzod Hakimov

Bielefeld University

3 PUBLICATIONS **7** CITATIONS

SEE PROFILE

Semih Yumusak

Karatay Univeristy

3 PUBLICATIONS **0** CITATIONS

SEE PROFILE

A Data-Model Driven Web Application Development Framework

Erdogan Dogdu
Georgia State University (adjunct)
TOBB University of Economics and Technology. Ankara, Turkey
edogdu@etu.edu.tr

Sherzod Hakimov
Bielefeld University
Bielefeld, Germany
shakimov@cit-ec.uni-bielefeld.de

Semih Yumusak
KTO Karatay University
Konya, Turkey
semih.yumusak@karatay.edu.tr

ABSTRACT
Model-driven approach for web application development is an important topic in software engineering. There are many existing tools to support model-driven engineering for web application development. However, most tools and techniques are complex and not very practical when it comes to real-life usage. Here we present a simple data model-driven approach for web application development that is based on RDF data model, the basic semantic Web data model, and its reasoning capabilities. We introduce a prototype implementation of the data model-driven Web application development framework that utilizes semantic Web technologies in the backend for the data model. In this framework, the design and development of the application is focused on the RDF data model and its reasoning logic (partially). Developers define the data model online using a Web application front-end and the framework generates different views of the data elements automatically. This enables the developers to change the data model easily whenever it is needed.

Categories and Subject Descriptors
D.2.6 [**Software Engineering**]: Programming Environments – *Integrated environments*. D.2.2 [**Software Engineering**]: Design Tools and Techniques. H.2.1 [**Database Management**]: Logical Design - Data models.

General Terms
Design, Experimentation.

Keywords
Model-driven engineering, semantic web for rapid application development, semantic web, software engineering.

1. INTRODUCTION
One of the major problems in engineering software systems for Web application is software maintenance, which is keeping software up-to-date with continued requirements changes after deployment. And most software systems, including Web applications, suffer from this problem.

Model-driven approach plays an important role in the development process of software systems. Many different techniques and languages are used for data modeling. Unified Modeling Language (UML) [1] is a well-known modeling language used in object-oriented software engineering. Recently, semantic Web technologies have been used successfully in some areas of software engineering specifically for model-driven engineering approaches. Rapid Web application development is a central topic in the software development world. Almost all Web application development frameworks are based on the well-known Model-View-Controller (MVC) design pattern. Data is in the "model" component, "view" component deals with the user interface of the application, and "controller" handles the business logic of the application, accessing data and sending the processed information to the user interface (view).

One of the disadvantages of the MVC pattern is the difficulty in software change management. Once a data model is designed and implemented, and the software operating on this data model is implemented, the data model and the controller software are very tightly linked. A change in the data model requires the control programs, which are using this model, to be changed as well. And this is a costly operation. Web application development frameworks, such as Ruby-on-Rails[1] or Yeoman[2] from Google, are not immune to this problem. These tools are very fast in the first phase of development for automatic user interface screen generations from data models. But later on any change on the data model, requires the developers to get into the code and make changes in the control and view software.

We address these problems by introducing a Web application development framework that we developed for rapid Web application development using RDF data-model driven approach. The data model is based on Resource Description Framework[3] (RDF) and all types of data (schema and instances) are stored and managed in an RDF triplestore[4].

The paper is organized as follows. Section 2 presents previous work on this and related topics. In section 3 we present RDF data model. Section 4 is about a motivating example "Unibase" which is a simple representation of university knowledge base built using our framework. Section 5 explains our framework in detail. In section 6 we conclude and present future work.

2. RELATED WORK
Semantic Web[5] technologies are developed in time to a set of standards for data modelling and querying for the Web of data. Ontologies are used in semantic Web, and in knowledgebase systems in general, as a rich data modelling tool. There are a lot

ACM SE '14, March 28 - 29 2014, Kennesaw, GA, USA
Copyright 2014 ACM 978-1-4503-2923-1/14/03□$15.00.
http://dx.doi.org/10.1145/2638404.2638522

[1] Ruby-on-Rails http://rubyonrails.org
[2] Yeoman, http://yeoman.io
[3] RDF, http://www.w3.org/RDF
[4] Triplestore, http://en.wikipedia.org/wiki/Triplestore
[5] http://www.w3.org/standards/semanticweb

of work in the literature on the utilization of semantic Web technologies and specifically ontologies in software engineering from a number of different aspects [2] [9]. Model-driven software engineering is a process of developing a model that describes the system and then transforming that model into executable form (e.g. source code). OWL 2 [3] is used in some proposed methods along with Ecore[1] (Eclipse Modeling Framework Project) [4] in [10]. Internet Application Modeling Language (IAML) is model-driven approach for Rich Internet Applications (RIA) supporting many features of web applications [11]. Bernstein provides some examples in [12] about semantic web technologies and software engineering. KOMMA is an ontology-driven application development framework that supports persistence for object triples by mapping to Java classes [13]. Recent work, such as Bergman's [5][6], suggest using ontologies as 'engines' for data-driven applications. Mapping relational databases to RDF data stores has been an area actively worked on recently due to progress in the semantic web area [7]. Bizer and Cyganiak implemented D2R Server which enables users to access relational model and data through Web. Users can query non-RDF databases using the SPARQL query language [8]. Ruby on Rails is an open source web application framework for constructing basic models, views and controllers for web applications. User can automatically and rapidly generate initial code for Web applications [1]. Eclipse Modeling Framework is a modeling framework for code generation for building structural applications based on data model [4]. Unified Modeling Language (UML) is a standardized modeling language used in software engineering for building object-oriented applications with data models [1]. There are CASE tools that take UML models and generate software automatically such as IBM Rational. Unlike the current ontology-based systems, we provide a generic framework that simplifies the Web application development process.

3. RDF DATA MODEL

We propose the use of a generic data model for Web applications that is based on the semantic Web data model standard, *Resource Description Framework* (RDF). RDF is a simple data model consisting of triple statements in the form of subject-predicate-object triples. For example, "University U1 has Department D1" is one such statement (or triple) where U1 is the subject, D1 is the object, and "has" is the predicate. This way all complicated relationships can be represented in RDF model by way of objects and the binary relationships between as in object-oriented model as iobjects.

RDF can be represented as a graph with "nodes", also called *resources*, and "directed links or relationships" between nodes, also called *properties*. Below is one such example that depicts the data elements in a university with objects and the relationships between them (Figure 1):

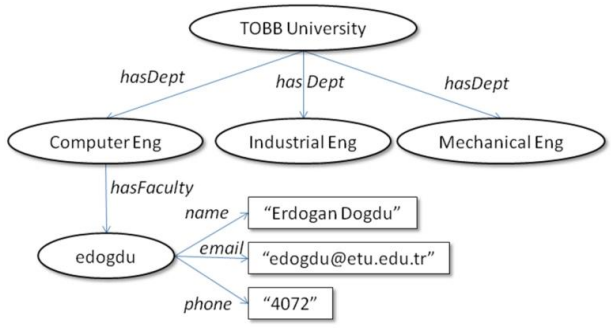

Figure 1. University data model in RDF as a graph

In this graph model, oval nodes represent objects such as "edogdu" (a faculty member in computer engineering department) and rectangular nodes represent literal values such as the phone number of the faculty edogdu (literal values are given in quotes). The literal value nodes are end points (not objects), they cannot be linked to other objects; only object nodes can be linked to the other object nodes or literal value nodes.

This data model, if transformed to a relational data model, is represented as relations or tables to define the entities such as universities, departments, faculty, etc. and the relationships between concepts such as "department d1 has faculty faculty f1" again in the form of a table.

Relational model requires a new table or relation for every new concept or entity or relationship between entities. Therefore, it is more complicated and more difficult to maintain. We on the other are proposing the use of RDF data model which requires only a single table or triple store to represent all entities and relationships. The data model in Figure 1 for example can be represented in an RDF triple store using triples shown in Table 1. This is an easier schema to develop, update, and maintain than the relational data model where the same example requires at least 4 relations (tables); university, department, faculty and department_faculty (in the case a faculty is cross-listed in multiple departments) tables.

Table 1. RDF triples for the data model in Figure 1.

Subject	Predicate	Object
TOBB University	hasDept	Computer Eng
TOBB University	hasDept	Industrial Eng
TOBB University	hasDept	Mechanical Eng
Computer Eng	hasFaculty	edogdu
edogdu	name	"Erdogan Dogdu"
edogdu	email	edogdu@etu.edu.tr
edogdu	phone	"4072"

4. A MOTIVATING EXAMPLE: UNIBASE

We developed a sample application, called Unibase, using our framework. This is a knowledgebase to keep, update, and manage data in a university information system. The knowledgebase includes information about all departments, courses, faculty, students and the relationships among these entities. For simplicity, we only developed a portion of the application to handle the data about the departments in a university and the courses offered by the departments.

In most web application development frameworks, the development process follows consists of these steps: (1) design the database model based on requirements, (2) design and implement the server side components, such as JavaBeans components and view components that generate user interfaces, (3) correct and enhance the user interface screens (4) test and repeat the process until a satisfactory application is developed. This is usually a long process involving people from different areas of architecting and developing software (database designer, server-side component programmers, user interface designers and developers, etc.).

Recently rapid application development frameworks are introduced to help ease this process; these frameworks aim to help the developers to implement web applications rapidly using generic tools. One well-known example is Ruby-on-Rails Web application development framework [1]. We aim a similar goal in this work; we want to be able to develop UniBase rapidly using a semantic Web-based infrastructure that will allow the user to develop the application easily without coding. User creates the

data model online and add new resources, update or delete resources dynamically. Data is stored in a triple store which is queried by Jena framework[8].

5. A DATA MODEL-DRIVEN FRAMEWORK FOR WEB APPLICATION DEVELOPMENT

Based on the simple RDF data model, we developed a Web application framework. This framework is used to develop a Web application rapidly. In its current form the framework provides a simple and generic Web interface to develop a Web application online. Future versions will be used to develop applications programmatically via an API.

The generic Web browser interface of the framework is used to do the following:

a. Creating/deleting/updating data element types (resource types),
b. Creating/deleting/updating data elements (data instances),
c. Creating/deleting/updating relationships between data elements (properties),
d. Creating/deleting/updating user interfaces (views).

The framework uses MySQL database system[9] with Jena API to store data types, data instances and relationships in the form of triples. UniBase application that is using the framework is a Web application to manage the data model and the data instances.

5.1 Data Element Types

In our framework, first, data element types are created via the Web interface as shown below (Figure 2). For example a faculty element type is created as below (University and Department types are created first in the example and shown on the left hand side of the screen):

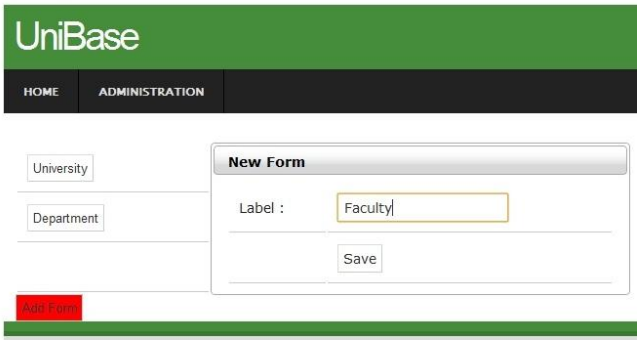

Figure 2. Create Data Element

5.2 Data Properties

Data elements are first represented by their data properties. For example a faculty member (data element) has data properties *name*, *email*, and *phone* as shown in the example above (Figure 1 and Table 2). The data values these properties have are literal values (therefore they cannot be linked to other data elements). This type of properties are "data properties" and they are marked as owl:DatatypeProperty properties as shown in Table 2. Besides the type of properties, the framework allows users to specify the label of the properties, and this is done by using rdfs:label property in RDFS. For our example, the labels of :name, :email and :phone properties are "Name", "Email" and "Phone". These labels are utilized in Web views presented to end users as prompts and list headers.

[8] http://jena.apache.org
[9] http://www.mysql.com

Table 2. All triples representing data properties

Subject	Predicate	Object
:name	rdf:type	owl:DatatypeProperty
:name	rdfs:label	"Name"
:name	:inListView	true
:email	rdf:type	owl:DatatypeProperty
:email	rdfs:label	"Email"
:email	:inListView	true
:phone	rdf:type	owl:DatatypeProperty
:phone	rdfs:label	"Phone"
:phone	:inListView	true

Data elements are viewed in two forms: (1) single instance view and (2) list view. In single instance view, the data properties of a single element instance are viewed in a web form; "rdfs:label" property values are shown on the left side of the form as prompts, and the corresponding data property values are shown on the right hand side. The list view on the other hand lists the data element instances as a list in the form of a table as shown below in Figure 3 (course elements).

Figure 3. List View of Data Elements

In the data model, properties that will be shown in the list view are marked with: inListView data property carrying a true value (Table 2). For example for faculty data element type, suppose that the data properties are name, email, phone, and office. And, suppose that only the first three are to be shown in the list view. Then, in our data model :name, :email, and :phone properties should have true value for :inListView property and :office property should have false value for :inListView property.

5.3 Object Properties

Data elements can also have object properties. Object properties are used to represent the relationships between data elements (objects). For example, a university has a number of departments; if universities and departments are separate objects, then :hasDept (has department) relationship between these objects is an object property (Figure 1).

To create, view and instantiate object properties, we again use the view module in our framework. User can create object properties in the data element type screen by pressing the "add relation" button. The, the user chooses the object type that the relationship points to, and the label of the relationship (Figure 4).

In this example "teaches" relationship between faculty and course element types is created. In the pop-up window user chooses the data element type that faculty "relates to" (course), enters the label of the relationship ('teaches'). There are two more options as seen in Figure 4, a checkbox to indicate whether the inverse relationship (between course and faculty) will be maintained in the database, and the label of the inverse relation ('taught by' in this case).

If an inverse object property is defined for an object property, then when instances of the object property are added, the inverse object property instances are automatically added to the data store.

This follows from the owl:inverseOf semantic reasoning rule in OWL as shown below[10] (T stands for triple):

```
T(?p1, owl:inverseOf, ?p2)
T(?x, ?p1, ?y)
T(?y, ?p2, ?x)
```

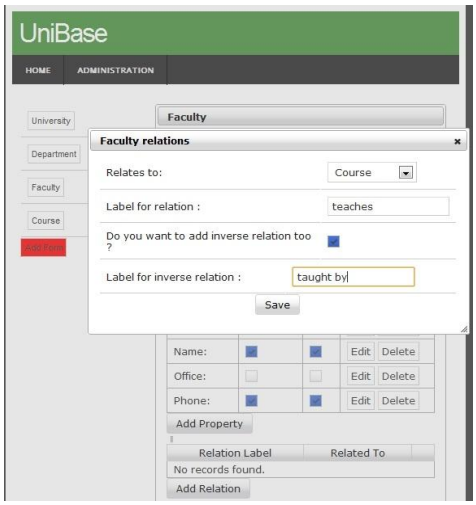

Figure 4. Adding a relation to a data element

For example, if edogdu-*teaches*-comp452 relationship instance is added, then comp452-*taughtBy*-edogdu instance is also added automatically to the data store by our framework for the above example as show below.

Subject	**Predicate**	**Object**
:edogdu	:teaches	:comp452
:comp452	**:taughtBy**	**:edogdu**

(inferred)

Eventually a data element (faculty) is viewed in the single instance view as shown in Figure 5 with its data properties (email, name, office and phone) and object properties (faculty-*teaches*-course) altogether. In this case the inverse relationship instance is also automatically added (course-*taughtby*-faculty).

Figure 5. Relation Instance

5.4 Change in Data Model

In our framework any change in the data model is handled dynamically. Any change on the data model that is entered via the user interface will be reflected in the triple store immediately and future operations will be affected. If a new data property is attached to a resource, the existing instances of that resource type, which are added before the change, do not have the new property. But the new instances that are added after the change will display the new property to be updated. For example, if course element type (Figure 3) is updated and a "website" data property is added, future course type instances will display the website property to be added in the single-instance view. If a property is deleted from a resource, all existing values for that property for the regarding resource instances will be deleted (not the resource but the property value attached to that resource). This is the cascading delete operation for the property values. In short, changes in the data model are instantly reflected in the existing data and affect all future operations.

6. CONCLUSION

RDF data model allows creation of more sophisticated and extensible data models with the support of ontology languages. Unlike the relational data model, RDF data model is stored as a set of triples (subject, predicate, object), and therefore change on the model is more flexible. Our framework currently handles only simple data-driven Web applications. There are many issues involved in developing a Web application and the framework we developed only addresses some of the main features needed to develop a full-fledged application. We are working on extending the framework.

7. REFERENCES

[1] Unified Modeling Language http://en.wikipedia.org/wiki/Unified_Modeling_Language

[2] Happel, H.-J., & Seedorf, S., "Applications of Ontologies in Software Engineering", In *Workshop on Semantic Web Enabled Software Engineering* (SWESE), 2006.

[3] *OWL 2* Web Ontology Language Document Overview http://www.w3.org/TR/owl2-overview

[4] Eclipse Modeling Framework Project (EMF) http://www.eclipse.org/modeling/emf/?project=emf

[5] Mike Bergman, "Ontologies as the 'Engine' for Data-Driven Applications", see: http://www.mkbergman.com/492/ontology-best- practices-for-data-driven-applications-part-3, 2011.

[6] Mike Bergman, "Ontology-Driven Apps Using Generic Applications", http://www.mkbergman.com/category/ontology-best-practices, 2011.

[7] "A Survey of Current Approaches for Mapping of Relational Databases to RDF", W3C RDB2RDF Report, 2009.

[8] Christian Bizer and Richard Cyganiak "D2R Server – Publishing Relational Databases on the Semantic Web", Proc. Of the 5th Int. Semantic Web Conference (ISWC), 2006.

[9] Achilleos, A., & Kapitsaki, G. (2011). A Model-Driven Framework for Developing Web Service Oriented Applications. *Model-driven Web Engineering Workshop*.

[10] Staab, S., Walter, T., Gröner, G., & Parreiras, F. (2010). Model driven engineering with ontology technologies. *Reasoning Web. Semantic Technologies for Software Eng.*, 62–98. Springer.

[11] Wright, J. M. The Development of a Modelling Language for Rich Internet Applications. *Engineering*, 2-5.

[12] Bernstein, A. (2011). Software engineering and the semantic web: a match made in heaven or in hell? *Software Language Engineering*, (200021), 203–205. Springer.

[13] Wenzel, Ken. "Ontology-Driven Application Architectures with KOMMA." *Workshop on Semantic Web Enabled Software Eng.* 2011.

[14] Ruby on Rails http://rubyonrails.org

[10] http://www.w3.org/TR/owl2-profiles/