Synth – Linked Data Application Implementation Environment

Mauricio Henrique de Souza Bomfim, Daniel Schwabe

[2]Informatics Department, PUC-Rio Rua Marques de Sao Vicente, 225.
Rio de Janeiro, RJ 22453-900, Brazil

mauriciobomfim@gmail.com, dschwabe@inf.puc-rio.br

Abstract. In this demo, we present Synth, a new development environment to implement Linked Data applications. It is best used in conjunction with the SHDM method, making it possible to take any RDF data available on the Linked Data cloud, extend it with one's own data, and provide a Web application that exposes and manipulates this data to perform a given set of tasks, including navigation, as well as general business logic.
Keywords: Linked Data, Semantic Web, Linked Data Applications, RDF, Design Method, Model Driven Development.

1 Introduction

The advent of the Semantic Web, and especially the more recent growth of the Linked Data initiative, has spurred the emergence of the Linked Open Data (LOD)[1] cloud, a collection of interlinked data sources spanning a wide range of subjects. It is now quite feasible to design Web applications (hosted in websites) whose contents are at least partially drawn from the LOD cloud. - e.g. BBC's 2010 World Cup website[2].

This new scenario has given rise to a new challenge – how to effectively build applications that consume linked data, often combining with locally generated and managed data – that we will call "Linked Data Applications", LDAs in short. These applications have as one of its basic tenets is the reuse of existing information – both vocabularies and instance data, whenever possible. They also follow the principle that the data should carry as much machine-processable semantics as possible, so it should be expected that, within feasible limits, the application semantics also be captured in machine processable way. Furthermore, to be consistent with the LDA philosophy, they should be specified using models expressed in the same formalisms used to describe the data itself.

In this demo we present Synth, a development environment that allows the specification of LDAs according to SHDM (Semantic Hypermedia Design Method) models [2], and the generation of running code from this specification. Using Synth, it is possible to generate LDAs with little or no programming, simply by declaring models as proposed by SHDM (see [1] for details). For reasons of space, we do not detail the example to be demonstrated here; see http://www.tecweb.inf.puc-rio.br/synth, where it can also be downloaded.

[1] http://linkeddata.org
[2] http://www.bbc.co.uk/blogs/bbcinternet/2010/07/bbc_world_ cup_2010_dynamic_sem.html

2 The Synth Development Environment

Synth is a development environment for building applications that are modeled according to SHDM. It provides a set of modules that receives, as input, models generated in each step of SHDM and produces, as output, the hypermedia application described by these models. Synth also provides an authoring environment that facilitates the adding and editing of these models through a GUI that can run on any web browser.

2.1 Software architecture

The software architecture of Synth was designed to be independent of its implementation technology. It consists of a set of modules, each responsible for maintaining and interpreting one of the models generated in each phase of SHDM. Each module is composed by a model described in a corresponding ontology in RDFS or OWL, and an interpreter that gives semantics to the models, in addition to the basic semantics of RDFS and OWL, in which they are represented. These modules work together, interpreting their models and communicating with each other, in order to generate the application runtime in accordance with the definitions of each model.

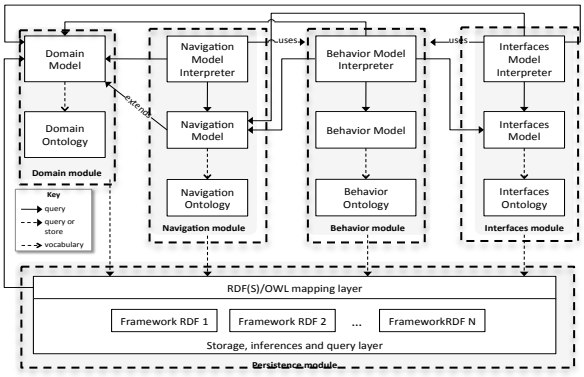

Figure 1 – Conceptual view of the Synth software architecture

Figure 1 shows a conceptual view of the modular software architecture for applications modeled using SHDM. The dashed boxes represent the modules and the white boxes represent the components of each module.

The persistence module handles the access and manipulation of application data, composed of two layers: a storage, inferences and query layer and a RDF(S)/OWL mapping layer. The former provides a single interface for accessing multiple environments and platforms for RDF data. It converts the access interfaces of various RDF data environments (e.g., Jena, Sesame, Virtuoso or OWLIM) into a single interface known by the RDF(S)/OWL mapping layer. This layer is also responsible for distributing queries among various data repositories, local or remote, combining their results, as enabled under the Linking Open Data initiative.

The RDF(S)/OWL mapping layer provides a view of the data and meta data, originally persisted as RDF triples, as primitives of the programming language in which the application is implemented.

The other modules are the domain, navigation, business logic, and interfaces modules. Each of these maintains and interprets its corresponding models, and is

similarly structured. The only exception is the domain module, which has no interpreter because the semantic of the domain model is that of RDFS and OWL, and the persistence module provides an interpreter for them in the RDF(S)/OWL mapping layer. Synth applications can read, write and create new RDF data in the LOD cloud.

2.2 Module collaboration

To illustrate how the modules collaborate, Figure 2 presents the sequence diagram showing the events in a typical execution of a "navigate" hypertextual navigation operation. The user interaction always happens through the external operations of the business logic model, available as Web Services, invoked by sending HTTP requests to the application. External operations fulfill the same role as the "controller" in the MVC-based (model-view-controller) architectures, coordinating user interactions, obtaining data from the domain Model, instructing the View to generate the interface and, finally, delivering the result to the user or external agent.

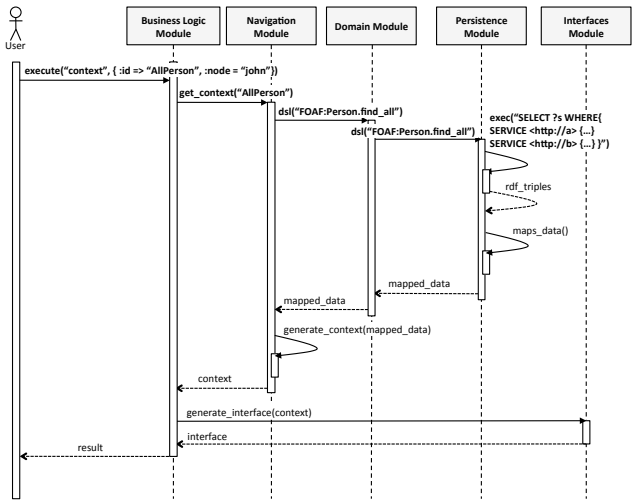

Figure 2 – Collaboration between modules in Synth

In this sequence diagram, the user sends a message to the business logic module invoking the method "execute" informing the name of an operation, in this case "context", and some parameters for this operation. This operation performs the "contextual navigation" operation of the SHDM hypertextual navigation model.

After that, the external operation invokes the method "get_context" from the navigation module with the context identifier "AllPerson" as a parameter. The navigation model retrieves the "AllPerson" context definition, obtaining the query expression. This expression is then used to invoke the method "dsl" from the domain module, which simply passes it to the persistence module. This module converts this expression to an equivalent Federated SPARQL query expression, executes it getting the results as RDF triples, and maps these results into the programming language primitives. Thus, Synth can access data in any SPARQL endpoint in the LOD cloud.

This mapped data is returned to the domain module and passed to the navigation module, which generates the context instance with its nodes based on the received data, and returns this context instance to the business logic module. Then, the business logic module invokes the method "generate_interface" in the interface

module passing the context instance as parameter, which returns the actual concrete interface. Finally, the business logic module returns this result to the user.

2.3 Implementation

Synth is implemented using Ruby on Rails, an MVC framework for web applications development. Within the MVC architecture, it maintains a modular organization according to the architecture shown in Figure 1.

All data in Synth is maintained as an RDF graph, including not only the instances of domain data, but also the metadata about the models and meta-models of SHDM, expressed as RDF(S) and OWL. This data is manipulated programmatically using the Sesame framework as an API. It plays the role of the storage, inference and query layer in the persistence module in conceptual architecture. The RDF(S)/OWL mapping layer is implemented using ActiveRDF [4] is a library for accessing RDF from Ruby programs, mapping the RDF data into Ruby primitives. ActiveRDF provides an API that facilitates CRUD operations within Ruby programs, and has an adapter for Sesame. Our previous experience (see [3]), and Oren et al [4], explains the reasons we have chosen Ruby and Rails as the implementation environment.

Synth provides an authoring environment GUI using HTML forms that can be accessed from a web browser and allows the creation and editing of primitives of SHDM models. It is possible run the application while it is being built using this interface, validating it in each step of the development process.

Synth also provides the RDF Scaffold, a generic RDF browser and editor that allows the execution of CRUD operations over the local application RDF database, in the same way that it can be done in some well known RDF browsers and wikis like Tabulator, Disco or OntoWiki.

3 Conclusion and Future Work

Future work is currently being carried out, adding new models to SHDM such a authorization, transactions, and adaptation, and their corresponding integration in Synth. Several optimizations are also being implemented, e.g., advanced caching.

Acknowledgement – The authors were partially supported by grants from CNPq and Petrobras.

References

1. Bomfim, M. de S.; Schwabe, D., Design and Implementation of Linked Data Applications Using SHDM and Synth. Proceedings of the Int. Conf. on Web Engineering ICWE 2011, LNCS 6757, Heidelberg, 2011, pp. 121-136.
2. Lima, F.; Schwabe, D.: “Application Modeling for the Semantic Web”, Proceedings of LA-Web 2003, Santiago, Chile, Nov. 2003. IEEE Press, pp. 93-102,
3. Nunes, D.A; Schwabe, D, Rapid prototyping of web applications combining domain specific languages and model driven design, Proc. 6th International Conference on Web Engineering (ICWE 2006), ACM, pp. 153-160. ISBN 1-59593-352-2
4. Oren, B. Heitmann, and S. Decker. ActiveRDF: embedding Semantic Web data into object-oriented languages. Journal of Web Semantics 6(3), 2008. pp 191-202.