Powl – A Web Based Platform for Collaborative Semantic Web Development

Sören Auer

University of Leipzig
auer@informatik.uni-leipzig.de

Abstract: We outline Powl, an opensource, web-based semantic web development platform for PHP. It may be used as a foundational framework for semantic web applications. Powl allows parsing, storing, querying, manipulating, versioning, serving and serializing RDF-S and OWL knowledge bases in a collaborative web enabled environment. Detailed project information, a demonstration and complete source code of Powl is available at http://powl.sf.net.

1 Introduction

In this demonstration paper we give an overview on Powl, a web based ontology management tool. Its capabilities include parsing, storing, querying, manipulating, versioning, serving and serializing RDF and OWL knowledge bases for different target audiences. For the semantic web researcher it may provide a test bed for rapid implementations of new approaches. For the web application developer it offers an object-oriented application programming interface for semantic web software development. For knowledge engineers there is a sophisticated web interface for collaborative development of ontologies in a web enabled environment.

Powl is implemented in the web scripting language PHP[1]. An implementation in a script language has serveral advantages. In addition to being platform independent, Powl is thus simple to modify and extend by researchers and semantic web developers. The source code is compact and easy to grasp. This results in a short development time - Powl was developed in approximately 9 man-months so far.

An advantage of PHP in particular is that it is by far the most widely used programming language for web applications. Conservatively estimated PHP is available at 35% of all websites worldwide and thus outnumbering all other web application technologies. Since the Semantic Web is considered to be an extension of the current World Wide Web, we are deeply convinced that the semantic web paradigm will only be successful in a broad perspective if there are applications and tools available tightly interacting with PHP.

Beside this being the main reason for developing Powl we reviewed existing ontology management tools. Michael Denny's "Ontology Tools Survey" [16] summarizes features of 94 ontology editors. None of them seems to combine a collaborative, web

1. [1] The PHP Group: PHP Homepage, http://www.php.net.

accessible user interface with native RDF-S and OWL handling. However we want to mention some tools which inspired Powl development in some. Protégé in conjunction with the OWL-Plugin [17] provides a highly advanced user interface for editing OWL knowledge bases. With Sesame [18] and Redland [19] efficient storage and querying engines are available. The Jena API [20] provides a highly comprehensive application programming interface for working with knowledge bases. Each of them has its particular focus. Powl is meant to be a comprehensive ontology management tool. It consistently integrates different facets of ontology management such as storage and querying, supplying an application programming interface and a collaborative web user interface. The price of generality, of course, is that Powl cannot be leading in every subarea of ontology management. Compared with other all-in-one ontology management tools like KAON [21] or WebODE [22] Powl tries to be in its core as close to the Semantic Web knowledge representation standards as possible.

2 Architecture

Powls architecture consists of 4 stacked tiers, while trying to minimize dependencies and supplying clean interfaces between tiers. It consists of the following tiers:

- *Powl store* – SQL compatible relational database backend
- *RDFAPI, RDFSAPI, OWLAPI* – layered APIs for handling RDF, RDF-Schema (RDFS) and OWL
- *Powl API* – containing classes and functions to build web applications on top of those APIs
- *User interface* – a set of PHP pages combining widgets provided by Powl API for accessing (browsing, viewing, editing) model data in a Powl store.

The generic API components are described in more detail in the remainder of this section, while more Powl specific functionality and the user interface are presented in Sections 3 to 7.

Powl Store
Any SQL compatible relational database supported by AdoDB [9], which is the database abstraction layer used by Powl, may act as a Powl store. The following database tables are used to store all information related to ontologies and their evolution:

Table	Description
`models`	provides information about the models in the store
`statements`	contains all statements of models in the store
`log_actions`	holds information about editing actions on a model
`log_statements`	contains added and removed statements for every action

The Powl store uses a denormalized database schema, where all resources and literals are written in full in a table row representing an RDF statement. Tests done by the RDFAPI developers state in [12] that this is 2 to 3 times faster than a normalized

database schema where resources and literal values are stored separately. The Powl store is accessed by RDFAPI.

RDFAPI
RAP – RDFAPI for PHP [12] – is an independent project by Chris Bizer, Radoslaw Oldakowski and others. It provides the following functionality to Powl:

- Parser, serializer for different RDF serializations (XML, N3, N-Triple)
- RDQL declarative query backend
- Classes and methods for working with RDF models, resources and literals
- NetAPI for publishing models on the web.

The higher layered APIs, RDFSAPI and OWLAPI, extend the classes "Model", "Resource" and "Literal" provided by RDFAPI.

RDFSAPI
RDFSAPI extends RDFAPI's class schema by RDF-Schema [8] specific classes. The interpreted languages approach of "typelessness" is applied, whenever a resource (class, instance, property respectively) is requested (e.g. as a function parameter) the following options representing the resource are available:

- RDFSResource object
- Local name (as a string, e.g. "Article")
- URI (as a string, e.g. "http://purl.org/net/nknouf/ns/bibtex#Article")
- Namespace prefix and local name (as a string, e.g. "bibtex:Article")

The following table gives an overview over methods exposed by the main RDFSAPI classes. Each value in curly brackets stands in conjunction with the prefix (as "add" or "list") for a different method. A complete API documentation including required parameters for each method and detailed explanations can be found at [1].

RDFSModel	RDFSClass
add{Triple,Class,Property,Instance} count{Triples,Classes,Properties,Instances} find{Triples,Resources} get{Triple,Resource,Class,Property,Instance} listTop{Classes,Properties} list{Resources,Classes,Properties,Instances} list{Namespaces,Languages,Datatypes} remove{Triple,Resource,Class,Property,Instance} load	add{Instance,Property} count{Instances,Subclasses} find{Instances} get{Instance} list{SubClasses,SuperClasses, Properties,Instances} remove{Instance,Property} set{SubClasses,SuperClasses, Properties}

RDFSProperty	RDFSInstance
add{Domain,Range}	addPropertyValue
add{SubProperty,SuperProperty}	get{Class,Classes}
get{Domain,Range}	get{PropertyValue}
list{Domain,Range}	listProperties
list{SubProperties,SuperProperties}	list{PropertyValues}
remove{Domain,Range}	setProperty{Value,Values}
set{Domain,Range}	

OWLAPI

All classes of RDFSAPI are extended with methods for handling OWL predefined properties, DL axioms and restrictions, as well as basic subsumption inference. Powl doesn't store entailed triples. In case of more abstract ontology languages layered on top of RDFS or OWL this would lead to an explosion of required database storage. Entailment is calculated when needed and cached for reuse.

3 Model Evolution and Versioning

To enable domain experts to collaboratively develop shared conceptualizations based on the Ontology Web Language a key requirement is to support a versioning strategy. Every editing action can be decomposed into smaller editing actions and finally into adds and removes of RDF triples to or from the RDF model. The following example illustrates this:

```
Class "owl:wine" updated
  Labels added
    Label for language "de" added: "Wein"
      Statement added:
        <owl:wine> <rdfs:label> "Wein"
    Label for language "ru" added: "Vino"
      Statement added:
        <owl:wine> <rdfs:label> "Vino"
  Annotation property
    "rdfs:seeAlso" changed from "wn-concept:103880346" into
                               "http://en.wikipedia.org/wiki/Wine"
      Statement removed:
        <owl:wine> <rdfs:seeAlso> "wn-concept:103880346"
      Statement added:
        <owl:wine> <rdfs:seeAlso>
                               "http://en.wikipedia.org/wiki/Wine"
```

Powl enables rollback of every particular editing action by determining if the involved triples are still present (if added) or still missing (if removed). A parent action thus may only be rolled back if all sub-actions may be rolled back as well.

4 Customizability by Software Knowledge

Instead of using configuration files or special database tables for customization, user authorization and preference management as well as user interface translation purposes and module integration, Powl uses a "system ontology" for storing such knowledge. This "system ontology" may be edited and managed using Powl itself. The user or administrator is further generically restricted to apply only valid configurations and this approach expands software flexibility dramatically. Three examples for knowledge in this system model are given:

- Instances of the system ontology class "Label" contain translations for all texts presented on the user interface and keep track of their usage in different parts of the application, thus simplifying the translation.
- Instances of the classes "User" and "Group" are responsible for authorization and storing of preferences.
- Further classes are used to store configuration data of Powl modules and widget plugins. Widgets for presenting and editing literal data may even be selected and configured dependent on the context they are actually used in (e.g. the user logged in, the data type of the literal or the property which the data value is assigned to).

Powl may be extended by software modules which are placed in a subfolder of the Powl distribution. Modules consist of:

- A list of namespaces for which the module should be active – if a resource of the actual model belongs to one of these namespaces the module will be loaded.
- Functions or PHP pages which provide a new view on the knowledge base or expose distinct functionality to be presented in a separate tab.
- Functions or PHP pages which may be called for an arbitrary triple, resource, class, property or instance – these will be linked with the corresponding objects in other modules whenever the user selects a triple, resource, class, property or instance.

For example a semantic web content management approach was implemented as a Powl module.

5 User Interface and Widgets for Data Editing

The user interface is arranged in tabs, each tab representing a different view on the knowledge base. The following tabs are included in Powl (additional ones may be added in a modular manner):

- *Models* – provides an overview on the models in Powls store.
- *Triples* – displays a browse-able and searchable list of the triples in the selected ontology.

- *Classes* – hierarchically organizes classes and allows viewing and editing their definitions.
- *Properties* – hierarchically organizes properties and allows viewing and editing their definitions.
- *Instances* – gives various views on instances of a distinct class in the model.
- *Version* – provides access to information about the evolution of the ontology

Powl provides a comprehensive library of plug-ins for comfortable editing of data:

- Single and multiple line text editing
- Radio, checkbox and drop-down widgets
- WYSIWIG HTML editing (HTMLArea component integrated)
- Selection or referencing of arbitrary resources from Powls store (may be restricted to classes, instances or properties).

The resource selection widget is equipped with "configurable intelligence": if only a few resources may be potentially selected radio, checkbox or drop-down widgets are used, otherwise browsing and searching is enabled in a separate window. All widgets may be customized by creating instances of a widget style profile class for the widget in the system ontology and connecting this with a suitable OWL property.

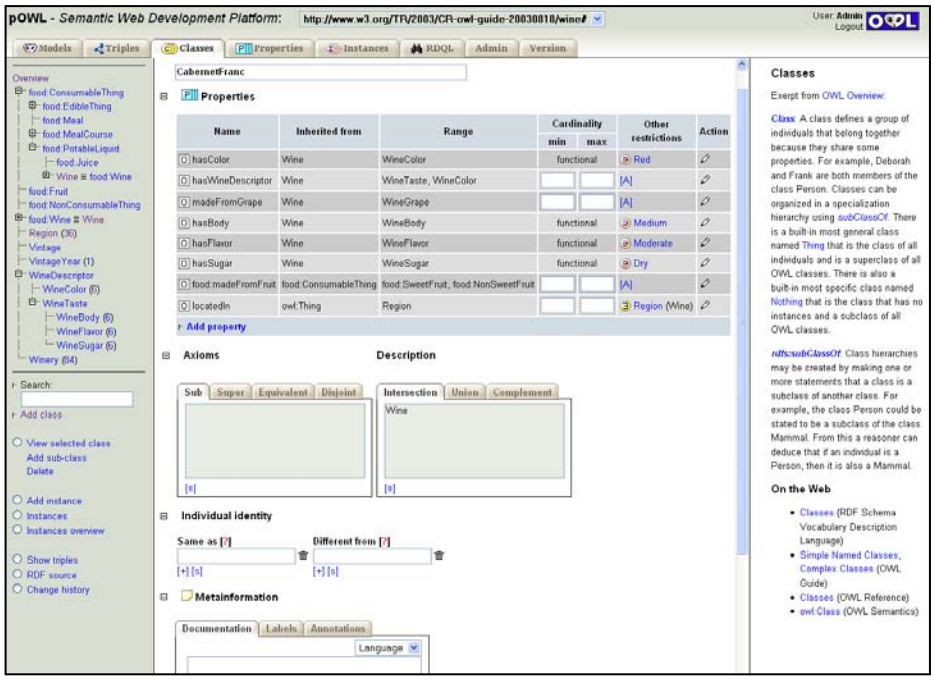
RDQL query builder
RDQL is an implementation of an SQL-like query language for RDF. It treats RDF as data and provides query with triple patterns and constraints over a single RDF model [14]. Powl enables users to freely formulate their queries or use the integrated query builder to question the knowledge base. It constraints the users input to only those values which make up a syntactically correct RDQL query, further more the user may only select resource and namespaces available in the knowledge base.

6 Different Points of View on a Model

From the programmer's as well as from the knowledge engineer's or user's point of view an OWL knowledge base may be seen through quite different glasses:

Triples View

Triples or RDF statements are related to natural language sentences consisting of subject, predicate and object. Subjects, predicates and objects may be resources – universally unique concept or entity identifiers. An object may furthermore consist of data typed literal values – literals. In RDFS and OWL such statements or RDF triples are used to define higher level objects like classes, properties and instances and establish relations between them. Powl supports this view on the knowledge base by enabling the user to view all RDF statements having a currently selected resource as subject, predicate or object. For the programmer RDFAPI provides an exhaustive set of methods to operate on triples.



Database View

An OWL knowledge base may be seen like an object-relational database. The following table establishes a informal correspondence between concepts in ontological knowledge representation and object-relational databases:

RDFS/OWL concept	RDBMS concept
Classes	Tables
Properties	Cols
Instances	Rows

To view a knowledge base in database manner, a highly configurable and filterable "instance overview" arranges instances of a class in a tabular view. Further instances may be exported and imported in spreadsheet and database compatible CSV format. The programmer is supplied with the SQL inspired declarative query language RDQL and methods for database table row like operations on instances.

Description Logic Axioms View

For the correspondance between Description Logic knowledge bases and OWL ontologies the Powl user interface provides special widgets supporting the user in viewing and editing class construction, property restriction and individual identity axioms. These widgets are bound to corresponding OWLAPI calls providing this functionality on the API level.

Serialization View

For RDF abstract syntax, which is the core of all models in Powl, there are different serialization formats: RDF/XML, N3, N-Triple. Powl supports viewing and editing of all parts of the ontology (classes, properties and instances with associated information) in these formats.

7 Scalability

Powl is designed to work with ontologies of arbitrary size (only limited by disk space). This requires that only those parts of the ontology are loaded into main memory which are required to display the information requested by the user on the screen (to render a web page containing this information). Special efforts are done to realize this for the subsumption tree view of classes. Subtrees are loaded on demand using Javascript and only if not already done. This mechanism even enables rendering virtually infinite trees in a webpage, which may occur if a subclass is declared to be equivalent to one of its super-classes.

The following benchmarks were taken on a recent system (Pentium 4, 2.8 Ghz, 512 MB RAM) with the WAMP (Windows, Apache, PHP, MySQL) software installed. The only optimization of the system was to enable MySQL's query cache. By further optimization (especially OPcode Caching as Zend Accelerator, APC or MMCache provide) significant speed improvements may be possible.

Model	Triple count	Import time	Classes	Class hierarchy calulation
Wordnet[2]	473 528	624 s	6	0.30 s
NCI Cancer Ontology[3]	463 878	597 s	27 652	0.46 s
UNSPSC[4]	82 500	82 s	16 499	1.06 s

[2] http://www.semanticweb.org/library/wordnet/
[3] http://www.mindswap.org/2003/CancerOntology/
[4] http://www.cs.vu.nl/~mcaklein/unspsc/

8 Conclusion

For domain experts and knowledge engineers, Powl provides an easy deployable and easy-to-use, web-based ontology editing and publishing solution. Powl supports collaborative work with ontologies as well as observing the ontology evolution. Powl is scalable and can be used even with extremely large knowledge bases. It may be customized according to specific needs. For PHP developers, Powl offers a comprehensive framework of functionalities for parsing, storing, querying, manipulating, serving and serializing RDFS and OWL ontologies. It may function as a basis for more domain specific web based ontology applications. Powl has been downloaded over 300 times since April 2004, when Powl development started. It is available under GNU Public Licence[5]. For future releases it is planned to integrate more powerful inferencing capabilities and to apply Powl for OWL-S [13] and web service development.

References

1. Auer, S.: Powl Homepage, The Web, 2004, http://powl.sourceforge.net/
2. Bechhofer, S., Dean, M., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D., Patel-Schneider, P., Stein, L.: Web Ontology Language (OWL) Reference version 1.0. Recommendation, W3C (2004), http://www.w3.org/TR/owl-ref/.
3. Beckett, D.: RDF/XML Syntax Specification (Revised). W3C. 10 February 2004.
4. Berners-Lee, T., Hendler, J., Lassila, O.: The semantic web. Scientific American 284 (2001) 35–43
5. Carroll, J., et. al: Jena: Implementing the Semantic Web Recomandations. Bristol. 2003. http://www.hpl.hp.com/techreports/2003/HPL-2003-146.pdf
6. Donini, F.M., Lenzerini, M., Nardi, D., , Schaerf, A.: Reasoning in description logics. In Brewka, G., ed.: Principles of Knowledge Representation. Studies in Logic, Language and Information. CSLI Publications, (1996) 193–238
7. Klyne, G., Carroll, J.: Resource Description Framework (RDF): Concepts and Abstract Syntax, W3C, 10 February 2004.
8. Lassila, O., Swick, R.R.: Resource description framework (RDF) Model and syntax specification. Recommendation, W3C, February 1999. http://www.w3.org/TR/1999/RECrdf -syntax-19990222
9. Lim, J.: ADOdb Library for PHP, http://php.weblogs.com/ADODB
10. McBride, B: "Jena: Implementing the RDF Model and Syntax Specification", in: Steffen Staab et al (eds.): "Proceedings of the Second International Workshop on the Semantic Web - SemWeb'2001", May 2001
11. Moore, G., Seaborne, A.: RDF Net API, W3C Member Submission, 2. October 2003. http://www.w3.org/Submission/2003/SUBM-rdf-netapi-20031002/
12. Oldakowski, R., Bizer, Chr.: RAP: RDF API for PHP, To be published in Proceedings of the "1st International Workshop on Interpreted Languages", 2004.
13. OWL Services Coalition: OWL-S: Semantic Markup for Web Services, The Web, 2004, http://www.daml.org/services/owl-s/1.0/
14. Seaborne, A.: RDQL - A Query Language for RDF, W3C Member Submission, 9 January 2004. http://www.w3.org/Submission/RDQL/
15. The PHP Group: PHP Homepage, http://www.php.net.

[5] http://www.gnu.org/copyleft/gpl.html

16. Denny, M.: Ontology Tools Survey, Revisited, 2004, http://www.xml.com/pub/a/2004/07/14/onto.html
17. Knublauch, H., Fergerson, R. W., Noy, N. F., Musen, M. A.: The Protégé OWL Plugin: An Open Development Environment for Semantic Web Applications, Third International Semantic Web Conference - ISWC 2004, Hiroshima, Japan (2004)
18. Broekstra1, J., Kampman1, A., Harmelen, F.: Sesame: A Generic Architecture for Storing and Querying RDF, Published at the International Semantic Web Conference 2002, Sardinia, Italy.
19. Beckett, D.: Redland rdf application framework. http://librdf.org/. ILRT, University of Bristol.
20. McBride, B.: Jena: Implementing the RDF Model and Syntax Specification, WWW2001.
21. Volz, R., Oberle, D., Staab, S., Motik, B.: KAON SERVER - A Semantic Web Management System, In Alternate Track Proceedings of the Twelfth International World Wide Web Conference, WWW2003, Budapest, Hungary, 20-24 May 2003. ACM, 2003.
22. Arpírez, JC.; Corcho, O.; Fernández-López, M.; Gómez-Pérez, A.: WebODE in a nutshell. AI Magazine 24(3):37-48. Fall 2003.