A General Web-oriented Architecture (WOA) for Structured Data

by Mike Bergman - Sunday, May 03, 2009

http://www.mkbergman.com/486/a-general-web-oriented-architecture-woa-for-structured-data/

An Abstract Web Services Framework Aids Broad Applicability

Structured Dynamics' product and software architecture is oriented to the Web. It emphasizes maximum flexibility, minimum "lock-in" and complete adaptability. This piece describes this architecture and how these aims are being met.

Design Objectives

Structured Dynamics is committed to what is known as a *Web-oriented architecture* (WOA) [1], which can be defined as:

WOA = SOA + WWW + REST

Nick Gall describes WOA as based on the architectural foundations of the Web, and is characterized by "globally linked, decentralized, and uniform intermediary processing of application state via self-describing messages."

WOA is a subset of the service-oriented architectural style, wherein discrete functions are packaged into modular and shareable elements ("services") that are made available in a distributed and loosely coupled manner. WOA uses the representational state transfer (REST) architectural style defined by Roy Fielding in his 2000 doctoral thesis; Fielding is also one of the principal authors of the Hypertext Transfer Protocol (HTTP) specification.

REST provides principles for how resources are defined and used and addressed with simple interfaces without additional messaging layers such as SOAP or RPC. The principles are couched within the framework of a generalized architectural style and are not limited to the Web, though they are a foundation to it.

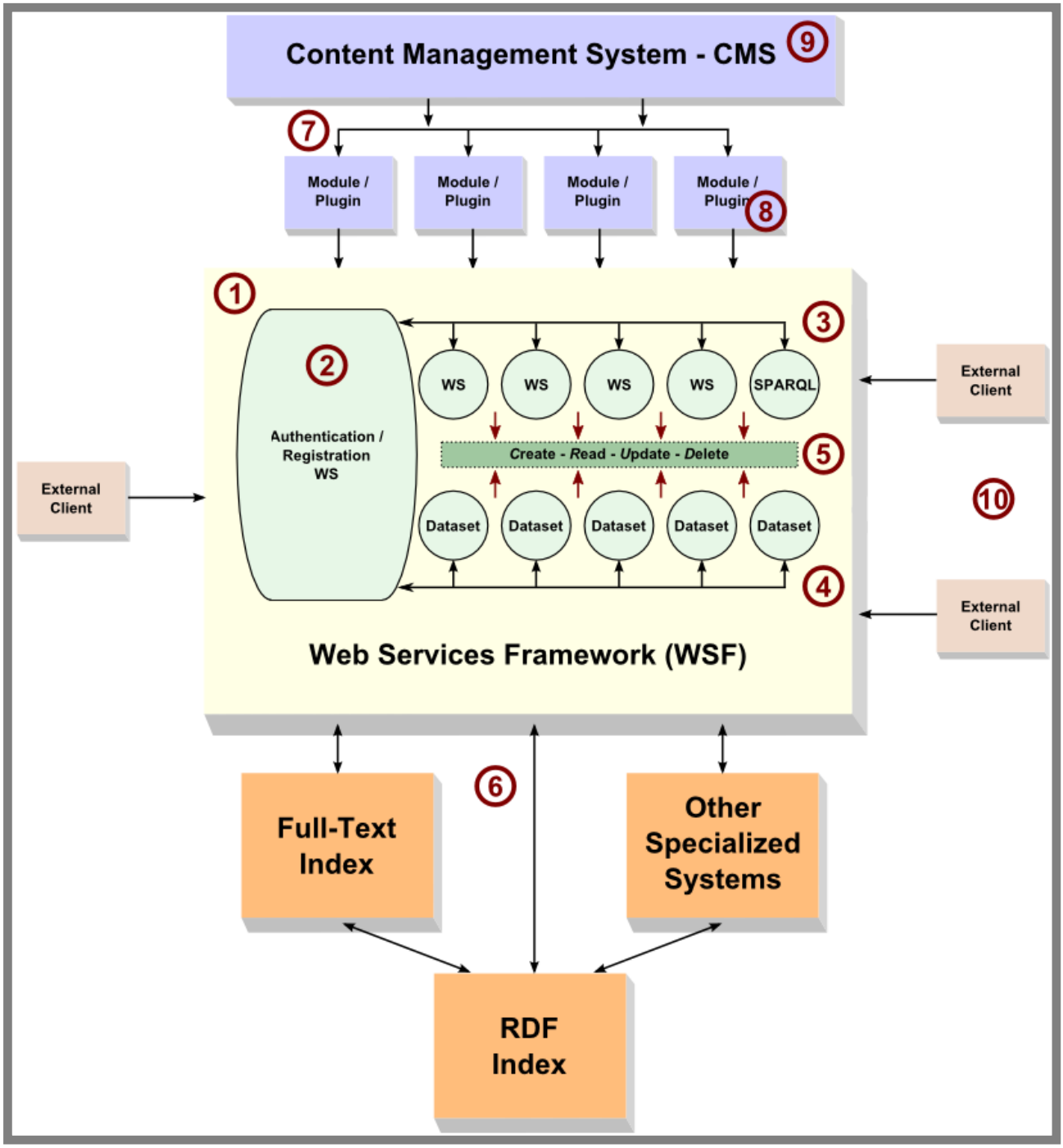
REST and WOA stand in contrast to earlier Web service styles that are often known by the WS-* acronym (such as WSDL, etc.). WOA has proven itself to be highly scalable and robust for decentralized users since all messages and interactions are self-contained (convey "state").

Structured Dynamics abstracts its WOA services into simple and compound ones (which are combinations of the simple). All Web services (WS) have uniform interfaces and conventions and share the error codes and standard functions of HTTP. We further extend the WOA definition and scope to include linked data, which is also RESTful. Thus, our WOA also sits atop an RDF (Resource Description Framework) database ("triple store") and full-text search engine.

These Web services then become the middleware interaction layer for general access and querying ("endpoints") and for tying in external software ("clients"), portals or content management systems (CMS). This design provides maximum flexibility, extensibility and substitutability of components.

Architecture & Components

Here is the basic overview of the architecture and its components. Each of its major components is described in turn as keyed by number:



The Web Services Framework Middleware

The core to the system is the Web services middleware layer, or WS framework (WSF). Structured Dynamics is preparing this framework [see **(1)**] as a separately available open source package under Apache 2 license (soon to be released). It provides all of the components shown as items **(2)** to **(5)** in the diagram.

WSF is the abstraction layer that provides the Web service endpoints and services for external use. It also provides the direct hooks into the underlying RDF triple stores and full-text search engines that drive these services. At initial release, these pre-configured hooks will be to the Virtuoso RDF triple store (via

ODBC, and later HTTP) and the Solr faceted text search engine (via HTTP) [2]. However, the design also allows other systems to be substituted if desired or for other specialized systems to be included (such as an analysis or advanced inference engine).

Authentication/Registration WS

The controlling Web service in WSF is the Authentication/Registration WS [see **(2)**]. The initial version uses registered IP addresses as the basis to grant access and privileges to datasets and functional Web services. Later versions may be expanded to include other authentication methods such as OpenID, keys (*a la* Amazon EC2), foaf+ssl or oauth. A secure channel (HTTPS, SSH) could also be included.

Other Core Web Services

The other core Web services provided with WSF are the CRUD functional services (*create* - *read* - *update* - *delete*), import and export, browse and search, and a basic templating system [see **(3)**]. These are viewed as core services for any structured dataset.

In initial release, the import and export formats will likely include TSV, RDF/XML, RDF/N3, RDF/Turtle, XML and possibly JSON.

Datasets, Access and Use Rights

A simple but elegant system guides access and use rights. First, every Web service is characterized as to whether it supports one or more of the CRUD actions. Second, each user is characterized as to whether they first have access rights to a dataset and, if they do, which of the CRUD permissions they have [see **(4, 5)**]. We can thus characterize the access and use protocol simply as A + CRUD.

Thereafter, a simple mapping of dataset access and CRUD rights determines whether a user sees a given dataset and what Web services ("tools") are presented to them and how they might manipulate that data. When expressed in standard user interfaces this leads to a simple contextual display of datasets and tools.

At the Web service layer, these access values are set parametrically. The system, however, is designed to more often be driven by user and group management at the CMS level via a lightweight plug-in or module layer.

A Structured Data Foundation

Fundamentally, this "data-driven application" works because of its structured data foundation [see **(6)**]. Structured Dynamics employs an innovative design that exposes all RDF and record aspects to full-text search and is able to present contextual ("faceted") results at all times in the interface [2]. In addition, the Virtuoso universal server provides RDF triple store and related structured data services.

The actual "driver" for the structured data system is one or more schema ("ontologies") setting all of these structured data conditions. These ontologies are also managed by the triple store. The definition of these ontologies is specified in such a way with accompanying documentation to enable new scopes or domains to easily drive the system.

Interactions with CMSs and External Clients

As described by the diagram so far, all interactions with the system have been mediated either by Web service APIs or external endpoints, such as SPARQL.

For external clients or any HTTP-accessible system [see **(10)**], this is sufficient. Programmatically, external clients (software) may readily interact with the WS and obtain results via parametric requests.

However, the framework is also designed to be embedded within existing content management systems (CMSs). For this purpose, an additional layer is provided.

The architecture of the system can support interactions with standard open source CMSs or app frameworks such as Ruby on Rails, Django, Joomla!, WordPress, Drupal or Liferay, as examples [see **(9)**].

CMS interaction first occurs via specific modules or plug-ins written for that system [see **(7)**]. These are very lightweight wrappers that conform to the registry and hooks of the host CMS system. The actual modules or plug-ins provided are also geared to the management style of the governing CMS and what it offers [see **(8)**]. Each module or plug-in wrapper is a packaging decision of how to bound the WSF Web services in a configuration appropriate to the parent CMS.

This design keeps the actual tie-ins to the CMS as a very thin wrapper layer, which can embrace an open source licensing basis consistent with the host CMS. Because all of the underlying functionality has been abstracted in the WSF framework, licensing integrity across all deployment layers is maintained while allowing broad CMS interoperability. The design also allows networks to be established of multiple portals or nodes with different CMSs, perfect for broad-scale collaboration. User choice and flexibility is retained to the max.

In this design, the CMS retains its prominence and visibility (and, indeed, the standard admin and licensing basis). The WSF, specific Web services, and structured data backend remain largely invisible to the user.

Benefits of the Design

This design has manifest benefits, some of which include:

- Broad suitability to a diversity of deployment architectures, operating systems and scales
- Substitutability of underlying triple stores and text engines
- Substitutability of preferred content management systems
- Access and use of Web service endpoints independent of CMS (external clients)
- Performant Web-oriented architecture (WOA) design
- Common, RESTful interface for all Web services and functions in the framework
- Easy registration of new Web services, inclusion with authorization system
- Ability to share and interoperate data independent of client CMSs or portals
- Use of the common *lingua franc*a of RDF and its general advantages [3].

A Note on Tools and Philosophy

Structured Dynamics has the twin philosophies of using the best tools available yet also to give its customers and clients full choice. For instance, SD believes the Virtuoso system to be the best RDF triple store with superior functionality. Our internal benchmarks affirm its performance. Virtuoso is our standard first recommendation for a performant triple store.

Yet our architecture and design is not dependent on this application, nor indeed any other application. Deployment environments, customer preference, or pre-existing installations sometimes warrant the use of certain tools or applications. Large collaboration networks necessarily spring from diversity. It is thus critical that SD's designs and architectures be tool-neutral and allow swapping and substitution. This is a major reason for the WOA and other RESTful design aspects of our Web services framework.

SD brings particular strengths in architecture, proper splits and design that separate ABox and TBox functionalilty [4], and ontology use and development. All of our designs are meant to be as tool neutral as possible, and we are always seeking the best of class in open source tools for any category.

Over the coming weeks and months Structured Dynamics will be rolling out packages and distribution sites for access to this framework and components built around this philosophy [5]. Stay tuned!

---

[1] See http://www.mkbergman.com/?cat=153.

[2] Frédérick Giasson, 2009. *RDF Aggregates and Full Text Search on Steroids with Solr*, April 29, 2009. See http://fgiasson.com/blog/index.php/2009/04/29/rdf-aggregates-and-full-text-search-on-steroids-with-solr/.

[3] Michael K. Bergman, 2009. *Advantages and Myths of RDF*, white paper from Structured Dynamics LLC, April 22, 2009, 13 pp. See http://www.mkbergman.com/wp-content/themes/ai3v2/files/2009Posts/Advantages_Myths_RDF_090422.pdf.

[4] As per our standard use:

"Description logics and their semantics traditionally split *concepts* and their relationships from the different treatment of *instances* and their attributes and roles, expressed as fact assertions. The concept split is known as the TBox (for *terminological* knowledge, the basis for *T* in *TBox*) and represents the schema or taxonomy of the domain at hand. The TBox is the structural and intensional component of conceptual relationships. The second split of instances is known as the ABox (for *assertions*, the basis for *A* in *ABox*) and describes the attributes of instances (and individuals), the roles between instances, and other assertions about instances regarding their class membership with the TBox concepts."

[5] This Structured Dynamics design has been aided in part as a result of research supported by NSF Award 0835851, Bibliographic Knowledge Network. The general Web services design and architecture is based on the system already developed for the UMBEL Web services.
