Fall 2008

LTER

01001100 01010100 01000101 01010010

LTER DataBits

Information Management Newsletter of The Long Term Ecological Research Network

01001100 01010100 01000101 01010010

Feature: Getting Started With Web Services

- Mason Kortz (PAL/CCE)

This article provides basic definitions and resources for developers starting to work with web services. It is an attempt both to organize the concepts I've learned over the last six months and to make the introduction to web services simpler for other LTER developers. It is far from a comprehensive guide to the development and use of web services. I encourage anyone with additional links, resources, or thoughts on web services to add to this article and add to our collective body of knowledge. To this end, I have started a thread on the LTER IM forum (link TBD) where additions to this article can be collected.

Terms and Definitions

These terms cover the basic language of web services. The definitions below focus on these terms as they are used in the context of web services; many of them have meaning in broader contexts. A far more complete (and more technical) set of definitions can be found at the W3C Web Services Glossary.

Service: An abstract interface to the functionality of an application through a set of standardized protocols. Services often provide machine-to-machine interface capabilities using a client-server model, but services may also communicate locally (i.e. communication between operating system components).

Web Service: A service that allows requests and responses to be exchanged over the Internet. Although web services do not, by definition, have to use the HTTP protocol, this is usually implied. In practical use, the term 'web service' is generally limited to services with responses intended to be used programmatically, rather than formatted for human readability by a browser.

Consumer: An application that sends requests to a web service and process the

responses.

Provider: An application that handles web service requests and provides responses.

SOA: Service Oriented Architecture. A modular system architecture in which applications provide functionality via autonomous, discoverable services. This allows complex processes to be performed by combining these services, rather that creating an application for each process.

SOAP: Simple Object Access Protocol. An XML protocol for exchanging messages between computers. These messages can be used to access web services in function-call-like manner. The consumer sends an XML request, which contains a function name and possibly parameters. The provider processes the request (i.e. calls the function) and returns an XML response. Both the request and the response are structured and strongly typed, meaning that each parameter and return value is given a type, such as integer, string, or array. New types may be defined by the combination of existing types, allowing complex data structures to be passed as parameters and return values.

REST: Representational State Transfer. An architecture for distributed hypermedia systems. In terms of web services, this refers to a service that accepts an HTTP request in the form of a subject (URI) and an action (GET, POST, PUT, or DELETE) and returns a representation of that subject in the resulting state. An important feature of a REST service is that the request contains all of the information needed to identify the resource and the state – there is no state information held in cookies or session variables. REST does not specify a structure for the request or response beyond the HTTP specification. Retrieving a static web page is a very basic REST operation.

WSDL: Web Service Description Language. An XML schema allowing the description of web services, including the URI of the web service, the functions it supports, the inputs and outputs of those functions, and the transport language used to communicate with the provider. With these specifications, a consumer application can automatically generate native calls to access the web service – for example, a PHP application could read a WSDL file and construct a PHP object with matching methods that would convert PHP calls into SOAP envelopes, and SOAP responses into PHP data structures. Theoretically, WSDL can be used to describe any web service regardless of transfer format and protocol, but in practice is most often used for SOAP services.

Developer Tools

Many languages have programming interfaces available to interact with web services by constructing SOAP envelopes and automatically prototyping functions based off of WSDL files. Desktop tools are also available for interacting with SOAP services and are very useful for viewing the XML level exchange between the consumer and the provider. REST interfaces and tools are fewer because of the lack of a description schema like WSDL, but various standards and code to support them are emerging.

Programming Interfaces

PHP: PHP4 supports SOAP through the PEAR::SOAP package. PHP5 provides similar support natively if the SOAP option enabled at compile. Both extensions can be used to write SOAP clients and servers. PHP can interface with REST services natively by opening URLs if the allow_url_fopen configuration is enabled.

PERL: SOAP and WSDL support are available through the SOAP::Lite package. This allows PERL scripts to act as both clients and servers in SOAP exchanges. REST services can be accessed using the HTTP functionality in the LWP package.

MATLAB: Current versions of MATLAB support SOAP transactions natively.

Python: Support for SOAP transactions and WSDL generation/interpretation are available through the SOAP.py package.

Other Tools

Google REST Describe/Compile – An experimental implementation of the WADL (Web Application Description Language) for defining REST web services. It's still in a demo state, but an interesting look at where REST services may be going.

Mac SOAP client: A lightweight desktop gui for exploring SOAP services.

Generic Web SOAP client: A web interface that provides basic SOAP service access.

Further Reading

There are a great number of excellent articles on web services available online, but separating the wheat from the chaff can be a little frustrating. The following articles were ones I found particularly informative, interesting, and enjoyable to read. For more information, the W3C web services home page and the Wikipedia

article on web services are both great starting points.

Architectural Styles and the Design of Network-based Software Architectures - Roy Thomas Fielding's dissertation, in which the term REST is coined and described. Pretty technical, but a great read on REST and network software architecture in general.

Automatic Multi Language Program Library Generation for REST APIs – An article by Thomas Steiner about the challenge of creating a WSDL-like description language for REST services, including overviews of a few proposed solutions.

How I Explained REST to My Wife – A very straightforward explanation of REST to a layperson, informative and easy to read.

The REST Dialogues and The REST Dialogues, A Real eBay Architect – A conversation between a developer and an imaginary eBay architect, and a response written by a real eBay architect. An easy introduction to the SOAP/REST debate.