Published on *LTER Databits* (http://databits.lternet.edu)

Web Services for ClimDB/HydroDB Database

Tue, 11/29/2011 - 17:19 — sremillard

- Good Tools And Programs

Issue:
Fall 2011

Suzanne Remillard (AND) and Kyle Kotwica

In the early years of the ClimDB/HydroDB database development, experimental web services were developed to access the database and to demonstrate potential capabilities and its interoperability with harvest scripts and other software. These scripts have never been developed further, but they are operational. This article is intended to provide an introduction to their capabilities, as well as a practical example of their use.

The application, dubbed the Web Service Administration Module for B2B Application Methods (WAMBAM), is a Perl tool developed to facilitate web service creation, providing a Graphical User Interface (GUI) that ClimDB/HydroDB Administrators can use to create, edit, and delete services. WAMBAM creates an XML configuration file that contains the necessary information to build any ClimDB service. These services can be one of three types; SQL, Perl or JAVA. In an SQL service, the configuration file lists the connection information, name of the method, input and output variables, and the query string to access the data in the database. In the case of Perl or JAVA, similar information is provided, but instead of SQL, executable code is submitted either inline or by submitting a path to the code.

Services created by WAMBAM are accessible through a SOAP Server, in this case a CGI application that listens for WAMBAM web service requests. This application also serves as an outwardly facing web page that provides documentation, an actual Web Service Documentation Language (WDSL) file, and a sample client for each of the services offered. The WSDL is an Extensible Markup Language (XML) file that provides all of the information necessary to create a client and to use the services. The sample clients created are really just links to publicly available third party software that inspects the WSDL and creates a web page that allows users to access or test the services through a browser interface. In practice, it is expected that end users would create clients that access the services.

The current web service examples can be viewed and tested here:
http://climhy.lternet.edu/wambam/soap_server.pl

When a request is made to a service, either through these generic browser clients or from a client on another machine, the request uses a handler (e.g., <service>_handler.pl) to receive the request and generate a response. Both the WSDL and the handler are virtual and are built on the fly with each call. The Soap Server listens for such requests.

Currently, two services have been built to access the ClimDB/HydroDB database. These are (1) climdb_raw, which accesses the raw data table, and (2) climdb_agg, which accesses the aggregated data table. Documentation can be found at
http://climhy.lternet.edu/climhy_ws_api.html

Climdb_raw currently includes the following methods:

get_day - returns all the data for a given research site id on a given day;
last_harvest - returns the date of the last harvest for a given a site, station, and a ClimDB long or short variable name;
get_variable - returns the value for a variable between a given date range.

Climdb_agg service currently includes the following methods:

get_agg_monthly - returns the aggregated data from climdb_raw for a given range of years;
get_agg_yearly - returns the yearly value of the aggregated variable over a given range of years;
agg_over_all - returns the monthly value of the aggregated variable over the entire period of record.

At the Andrews LTER, we incorporate the use of these web services as a way of streamlining ClimDB/HydroDB harvests. This approach, detailed below, is an example of how these ClimDB/HydroDB web services can be used today. To update the Andrews ClimDB/HydroDB data, we use an automated process that requests the last date of data in the database for a given station and variable via the last_harvest methods of climdb_raw service. The following example shows our client, which is wrapped in the subroutine 'get_start' called with a site, station, and variable combination that we use in our harvest application to make a call to the climdb_raw web service:

```
$start = get_start('AND',$station,$name);
sub get_start {
        my $start = SOAP::Lite
        ->uri("urn:climdb_raw")
        ->proxy("http://climhy.lternet.edu/wambam/services/climdb_raw_handler.pl")
        ->last_harvest(@_)
        ->valueof('///date');
        }
```

The client includes the web service name, the handler (one could use the WSDL here), the method, and the return variable(s). Although the handler is built on the fly whenever the WSDL

is inspected and will reflect any changes made to the service after it is deployed, the handler code is left on the server. Therefore it is possible to bypass the WSDL and call the service directly. This is the approach we use in our application. The value returned from this request is obtained from ClimDB/HydroDB database using the last_harvest method and then is used as a start date to build our ClimDB/HydroDB exchange file. Our exchange file contains a separate header line for each site, station, variable combination. Although it is easy enough to create and harvest a large exchange file with all the station data in it, this web service enables us to easily customize each data harvest with only the newest data and creates a much smaller file that is more efficient to harvest into the ClimDB/HydroDB database.

Thanks to Don Henshaw and John Chamblee for editorial comments on this article