

Re: Creating db front end or middleware.

Source: <http://coding.derkeiler.com/Archive/Python/comp.lang.python/2006-11/msg00712.html>

- *From:* Bruno Desthuilliers <onurb@xxxxxxxxxxxx>
 - *Date:* Mon, 06 Nov 2006 11:19:47 +0100
-

tobiah wrote:

Let's say I want to write a new tool to do something to, or report on people in a database. Each tool is going to have to have all sorts of routines that know about the relationship between the data. The first thought is to write a library of routines that do things like, `change_address()`, or `fire_employee()` or whatever. Whoops, now we've decided to move to python. I have to write all of that again.

So I thought, shouldn't there be a cloud that sits in front of the database to which I can ask these services of? Some process that awaits requests for information, or handles new incoming data. Now new apps written in any language should be able to use a basic protocol in order to get the work done.

The question then, is what the best way to do this is.

The first question then is: do you really need a middleware ? Most RDBMSs are already full-blown server applications – with support for authentication, permissions, triggers and stored procedures – that can be accessed by client programs in almost any language.

Now if you effectively want/need a web service, it seems that the canonical solutions are XMLRPC and SOAP. But if you already use a RDBMS, this web service should IMHO mostly be designed as an higher-level interface to the RDBMS, not as a full-blown app managing domain/business rules (which is the job of the RDBMS). IOW, it should be still possible for applications to directly interact with the RDBMS.

First I thought of using cherrypy to sit and listen to POST requests, but this would make passing complex structures extremely inconvenient.

Re: Creating db front end or middleware.

There are XMLRPC packages for most languages, that knows how to do the native data structure <-> XMLRPC translation. Python has both client and server packages in the standard lib.

Then I thought about WSDL.

Would this be the next logical step, or is this more for allowing different companies to communicate with each other when there needs to be a broadcast description of the interface?

As the name implies, W(eb) S(ervice) D(escription) L(anguage) is meant to describe a given web service – and says nothing about implementation.

My 2 cents

—

bruno desthuilliers

python -c "print