Re: Python gets macros − now XML does too

Source: http://coding.derkeiler.com/Archive/Lisp/comp.lang.lisp/2005−01/0195.html

---

From: Manuel Lemos (*mlemos_at_acm.org*)
Date: 01/04/05

Date: Tue, 04 Jan 2005 20:09:00 −0200

Hello,

on 12/18/2004 08:08 PM Frank Buss said the following:
> I think the main advantage: You use Lisp within a Lisp macro to generate
> Lisp code. I don't know how your system handles this (this was one reason,
> why I asked you to provide a translation of my simple example, to learn how
> your system works), but I think it is cumbersome to implement the code
> generation part in another language than the description to generate the
> code.

MetaL compiler flow module provides support for level 0 (macro expansion) metaprogramming but IMO that is a very limited form of metaprogramming.

Unless I misunderstood anything about LISP macros, it is not even unique to LISP. C has support for macros via the preprocessor, C++ has template classes and Java now has generics. There is no point in comparing the capabilities of each language macro expansion capabilities because that is not the topic here.

What is more interesting about the metaprogramming capabilities provided by MetaL is not metaprogramming features of level 0 (macro expansion) or even level 1 (generate code in other languages).

The real power comes from level 2 metaprogramming features and eventually beyhond. Level 2 means that you write a code specification in an high level syntax. That that specification is translated into MetaL level 1/level 0 XML source code, which gets translated to the final target language of choice. Currently PHP, Java, and Perl are supported but it could be also LISP.

Currently the only level 2 module of MetaL compiler that was develop is for generating persistence layers, i.e. classes that retrieve, manipulate store objects that are kept in SQL databases or eventually other persistence storage containers.

Personally I am using this persistence module to develop database applications with much more productivity as such applications would take

much more time to write, test and debug if they were written by hand.

Despite currently I only use PHP because I only develop Web applications, I don't know about the future. If I need to switch to another target language, I just need to develop the lower level translation modules of MetaL to support another language, and so my current work supporting PHP applications will not be wasted. This is the main goal of MetaL.

−−
Regards,
Manuel Lemos
PHP Classes − Free ready to use OOP components written in PHP
http://www.phpclasses.org/
PHP Reviews − Reviews of PHP books and other products
http://www.phpclasses.org/reviews/
Metastorage − Data object relational mapping layer generator
http://www.meta−language.net/metastorage.html