
Subject: Re: user defined idl_catalog.xml
Posted by [JD Smith](#) on Mon, 25 Sep 2006 17:21:45 GMT
[View Forum Message](#) <> [Reply to Message](#)

On Fri, 22 Sep 2006 21:07:53 -0600, Michael Galloy wrote:

> JD Smith wrote:
>> Interesting... any screenshots of an example? Don't you need an "ADP"
>> file as well (the IDL_Assistant wants one).
>
> Yes, IDLdoc will create the ADP file as well (using the ASSISTANT
> keyword makes IDLdoc create the ADP file and use a simpler HTML that the
> IDL Assistant can understand).
>
>> Right now, IDLWAVE simply parses that XML file, and then makes a LISP
>> equivalent in ~/.idlwave/idl_xml_rinfo.el for quicker re-load.
>>
>> If you provide source code along with your package, I have thought of
>> creating a special tag in the documentation header, along the lines
>> of:
>>
>> ;; %%%HTML-HELP%%% /path/to/html/help/file.html
>>
>> or some such, which can be loaded into the assistant if found. This
>> doesn't help if you distribute binary flavors of your routines,
>> however. The XML file itself is fairly straightforward to duplicate.
>> Aside from special cases like aliases (for those space-saving
>> documentation entries like OPENU/OPENR/OPENW), and system variables,
>> etc., the format is pretty simple:
>>
>> <ROUTINE name="WV_APPLET" link="WV_APPLET.html">
>> <SYNTAX name="WV_APPLET [, Input]" type="pro" />
>> <ARGUMENT name="Input" link="WV_APPLET.html#wp1004581" />
>> <KEYWORD name="ARRAY" link="WV_APPLET.html#wp1004583" />
>> <KEYWORD name="GROUP_LEADER" link="WV_APPLET.html#wp1004841" />
>> <KEYWORD name="NO_SPLASH" link="WV_APPLET.html#wp1004848" />
>> <KEYWORD name="TOOLS" link="WV_APPLET.html#wp1004856" />
>> <KEYWORD name="WAVELETS" link="WV_APPLET.html#wp1005269" />
>> </ROUTINE>
>>
>> i.e., name the routine, give the link (here assumed to be beneath
>> \$IDL_PATH/help/online_help), give syntax with (important) type="pro"
>> or type="func", list the arguments and their anchor links (if they
>> have separate links in the file), and similarly with keywords.
>>
>> Class methods are slightly different -- see the xml file.
>
> This XML file looks like it would be pretty easy for IDLdoc to create.

> IDLdoc has all that information at hand when it is creating its output.
> (It doesn't look all that different than the ADP file.)

Sounds great. Keep in mind that RSI/ITTVIS controls the format of this file, but I highly doubt they would change it (since it would break IDLWAVE). There are lots of corner cases in there, but the combination of <ROUTINE/> <CLASS/> <METHOD/> should suffice.

>> However, even if you create such a file, there is no simple way to
>> append the data to the one shipped with IDL. IDLWAVE assumes the two
>> sources of routine info are separate (library catalogs vs. system
>> routine info).

>>

>> If IDLDOC simply generates a webpage without lots of internal links
>> (for keywords, etc.), it would be simpler to just associate that html
>> file somehow with the source code through some markup convention.
>> Then rather than using "in source" help for library files, it could
>> call the assistant for display. We'd also need a convention for where
>> the IDLDOC html files are stored relative to the .pro file, e.g. just
>> specify relative directories:

>>

>> ;; %%%HTML-HELP%%% ../help/myroutine.html

>>

>> Perhaps Michael could overload the IDLDOC syntax to include something
>> like this, assuming that fits in logically. If it were trivial to
>> unambiguously recognize a block of IDLDOC markup (which it doesn't
>> appear to be as of IDLDOC 2.0), IDLWAVE (or, rather, idlwave_catalog,
>> the perl script which created library catalog files) could look for a
>> specific tag, like:

>>

>> ; @help_location ../help/myroutine.html

>>

>> save this path to the HTML file along with other information on the
>> routine (for simplicity, likely the next routine it finds in the
>> file), and then direct all help requests for that routine to the IDL
>> Assistant (or other configured web browser).

>>

>> That would be more straightforward than going the full XML route.
>> What that doesn't give you is the ability to jump right to the
>> documentation of a given keyword. E.g. try:

>>

>> IDL> widget_control, b, SCR_XSIZE=[M-?]

>>

>> I don't know if IDLDOC would ever provide that granularity.

>

> So if IDLdoc could write the XML file for the documentation and IDLWAVE
> knew the location of that file, then documentation for the user routines

> would have all the same functionality as the IDL documentation? That
> would be cool. I've been using IDLWAVE a lot more recently (and thanks
> for the M-? tip, I didn't know that) and that would be very useful. I'm
> not so sure about the @help_location tag since the documentation is
> meant to be independent of the code. (You would need a @help_location in
> each file? Then moving the documentation could mean a lot of edits.)
> Maybe IDLWAVE could keep a list of user library XML files? That would
> also mean that the source code for the user library wouldn't be needed.

Right now IDLWAVE only anticipates loading and caching a single XML file (the one provided with IDL), but it would be fairly easy to extend the framework to look for user-available ones. The question is, how to find them, and then how to locate the HTML/ADP files they reference? You'll note that the syntax doesn't say anything about where the files are:

```
<ROUTINE name="OPLOT" link="OPLOT.html">
```

IDLWAVE simply assumes they are in !DIR/help/online_help. For random user XML catalogs, the easiest might be some convention, perhaps an additional tag in the XML file, which specifies the default location to root the selected files. Perhaps IDLDoc even has a de facto convention of this sort already (I haven't used it yet)?

The other place we could hook this type of information in would be the "user catalog", which you may have seen as a bunch of .idlwave_catalog files in directories (e.g. AstroLib, etc.). As I alluded, this would be easy to do if there was only one URL associated with a given routine (e.g. no separate URL for keywords, etc.). Then we'd just need to tag that in some way in the file (either with the IDLDoc, or separately), and let idlwave_catalog strip that out. This is nice in a sense because (from IDLWAVE's point of view), the .idlwave_catalog file(s) would then be one stop shopping for routine and associated help info.

Either of these routes requires some work, but I think would be very useful in the long run.

JD
