
Subject: Re: taking the widget plunge. help

Posted by [Martin Schultz](#) on Tue, 12 Sep 2000 16:24:30 GMT

[View Forum Message](#) <> [Reply to Message](#)

"J.D. Smith" wrote:

>

> Martin Schultz wrote:

>

>>

>> This seems somewhat "convoluted" to me (but after recent experience, I

>> am sure you will have your reasons for proposing exactly this). I

>> always tend to think that setup is best done with ASCII files that are

>> easily editable and human readable. Yes, you should have a method

>> named something like FSC_PsConfig::Setup, and this method should

>> define a minimal set of defaults. But then it would read a file and

>> overload the default definitions. If it doesn't find the file, well,

>> then you live with the defaults (or the company creates a child object

>> with specific defaults). Proposed strategy:

> <snip>

>> As for the file format you could do something like

>> A4:

>> size=11.9,6.2 # not sure about the values

>> color=1

>> END

>>

>> A4_Landscape:

>> size=6.2,11.9

>> color=0

>> END

>>

>

> The problem with using a text file for the input, is that it's deceptively

> appealing. Easy to edit, no object knowledge required, etc. But, once you've

> set the format, you're basically locked into it. Want to add some new items or

> reorganize (for instance, making groups of setups)? You'll need special code to

> handle older-format input files (though you could obviously plan ahead for such

> contingencies). Want to reorganize the internal representation of the data

> entirely? You'll still have to accomodate the old input mechanism. For this

> problem, a flat-file input is probably tractable, but I thought it would be a

> good example case for maximizing forward compatibility. Backward compatibility

> is easy, if tedious. Forward compatibility (being able to replace aging

> modules/objects with new ones without changing the including code), is more

> troublesome.

>

> The idea of abstracting the interface to be limited to a defined set of methods

> with given arguments constrains the fixed interface specification to elements

> enforced by the language itself... certainly RSI won't change the meaning of

> arguments or remove keyword functionality. This abstraction is certainly

> sometimes overkill, as it is not without its costs. But for something which is
> intended to be upgradeable and extensible, I think it can be worth it.
>
> JD
>
> --
> J.D. Smith /*\ WORK: (607) 255-6263
> Cornell University Dept. of Astronomy */ (607) 255-5842
> 304 Space Sciences Bldg. /*\ FAX: (607) 255-5875
> Ithaca, NY 14853 */

I see. Yet, I would like to argue for text based setup files, because they are relatively easy maintainable and human readable. If I compare the Windows registry with the Unix ASCII based setup files, I certainly prefer the latter (for example one can use grep to find something, and one can add comments). With a format like the one I indicated, you still guarantee a lot of up- and downward compatibility: Variables that are undefined in one version will simply produce a warning message but otherwise be ignored. BTW: I don't agree that backward compatibility is always easy - at least not, if you are dealing with binary files of any kind (including IDL sav files).

I do concede, however, that ASCII files bear the danger of getting messy and "incompatible". To some extent, this can be solved by setting some standards such as:

- comments begin with '#'
- definition sections begin with a name followed by ':' and end with 'END'
- a setup file contains either no definition sections (i.e. variables are defined directly or "globally") or only definition sections
- each variable definition must contain a '=' even if the definition is empty

and to facilitate the search for the file:

- the filename of the setup file is <programname>.setup

One advantage compared to the sort of "internal" setup you are proposing is that even people who don't know much about IDL can customize the program (errr, the object ;-) whereas you need to know how to inherit if you want to overwrite a setup method. Well, on the other hand, this would give even more power to skilled programmers who can get rich ...

Cheers,
Martin

--

```

[[ Dr. Martin Schultz  Max-Planck-Institut fuer Meteorologie  [[
[[      Bundesstr. 55, 20146 Hamburg      [[
[[      phone: +49 40 41173-308      [[
[[      fax:  +49 40 41173-298      [[
[[ martin.schultz@dkrz.de      [[
[[

```
