Subject: Re: Custom formatting under version 7.0?
Posted by Michael Galloy on Thu, 06 Dec 2007 20:46:56 GMT
View Forum Message <> Reply to Message

On Dec 6, 1:21 pm, Scott Bolin <sboli...@gmail.com> wrote:
> On Dec 6, 11:38 am, "mgal...@gmail.com" <mgal...@gmail.com> wrote:
>
>> On Dec 6, 9:54 am, "Jim Pendleton, ITT Visual Information Solutions"
>
>> <jimp@no_spam.ittvis.com> wrote:
>>> To follow up on Doug's comment, creating templates for IDLdoc-style
>>> comments also helps discourage laziness in documenting one's code.
>>> Now if only I could attach keyboard accelerators to individual template
>>> insertion commands...
>
>> I've created some templates for writing classes/subclass (getProperty,
>> setProperty, cleanup, init, and __define, plus comment headers). Very
>> handy. My one issue is that it indents every line, so I have to select
>> all and move left. I don't see a way to fix that (and ITT VIS's
>> function and procedure templates do it too).
>
>> Mike
>> --www.michaelgalloy.com
>> Tech-X Corporation
>> Software Developer II
>
> Mike,
>
>   It should insert the template at the current cursor position. If
> you are in column 1 of the editor when you select CTRL+SPACE, and then
> pick your template, it should insert it beginning in column 1. This is
> not the behavior you are seeing?

OK. It's only one use case (which was the one I was using all the
time). Open a new file. Start on the first column of the first row.
Hit content assist. The list of choices will come up. *Type*
"subclass" (for my SUBCLASS template). Hit enter to select. Everything
is indented two spaces. If instead of typing "subclass", I down arrow
to "SUBCLASS" and hit enter, it is fine. Actually, the "PROCEDURE" and
"FUNCTION" templates seem to behave just fine, my problem there was
what you said -- I was already indented two spaces when I tried them.

My class template is shown below.

Mike
--
www.michaelgalloy.com
Tech-X Corporation

Software Developer

```
; docformat = 'rst'

;+
; Get properties.
;-
pro ${classname}::getProperty, _ref_extra=e
  compile_opt strictarr

  if (n_elements(e) gt 0) then begin
    self->${subclassname}::getProperty, _strict_extra=e
  endif
end


;+
; Set properties.
;-
pro ${classname}::setProperty, _ref_extra=e
  compile_opt strictarr

  if (n_elements(e) gt 0) then begin
    self->${subclassname}::setProperty, _strict_extra=e
  endif
end


;+
; Free resources.
;-
pro ${classname}::cleanup
  compile_opt strictarr

  self->${subclassname}::cleanup
end


;+
; Create ${classname} object.
;
; :Returns: 1 for success, 0 for failure
;-
function ${classname}::init
  compile_opt strictarr

  if (~self->${subclassname}::init()) then return, 0
  return, 1
end


;+
; Define instance variables.
;
; :Fields:
;
;-
pro ${classname}__define
  compile_opt strictarr

  define = { ${classname}, inherits ${subclassname}, ${cursor}}
end
```