Subject: Re: creating documentation file
Posted by Antonio Santiago on Tue, 15 Mar 2005 16:14:23 GMT
View Forum Message <> Reply to Message

I created my own perl script:
http://asantiago.gentelibre.org/index.php/archives/2004/12/0
1/a-perlscript-for-idl-documentation-generator/

but I change to IDLdoc when I found it:
http://asantiago.gentelibre.org/index.php/archives/2005/03/0 1/idldoc-documentation-utility/

Also attach a file that use IDLdoc syntax.

```
;+
; @file_comments Product class represents a radar product to
; generate. Maintains associations with the plugins which is based
; plugin information and parent dependencies.
;
; @author Antonio Santiago
;         (santiago\@grahi.upc.edu - http://asantiago.gentelibre.org)
;
; @requires linkedlist logmsg
;
; @field parents Linkedlist of products that represnts parent or input
;                dependencies
; @field children Linkedlist of products that represent children
;                 dependencies.
; @field executed Indicates if the product has been generated.
; @field name Name of the product.
; @field description Description of the product.
; @field plugin Plugin object that implements the product.
; @field plugin_info PluginInfo object that has the plugin
;                    information.
;
; @history
;
;       Fri Jan 14 17:08:04 2005, Antonio Santiago
;       (santiago\@grahi.upc.edu - http://asantiago.gentelibre.org)
;-
PRO Product__define
    struct = { Product, $
      parents: OBJ_NEW(), $
      children: OBJ_NEW(), $
```

```
      executed: 0B, $

      name: '', $
      description: '', $

      plugin: OBJ_NEW(), $      ;'Plugin' object
      plugin_info: OBJ_NEW() $  ;'PluginInfo' object
      }
END


;+
; Creates and initializes the object.
;
; @returns 1 if successful; 0 otherwise
;
; @private
;-
FUNCTION Product::Init
   self.parents = OBJ_NEW('linkedlist')
   self.children = OBJ_NEW('linkedlist')
   RETURN, 1
END


;+
; Frees the resources used by the object.
;
; @private
;-
PRO Product::Cleanup
   ;;Frees Plugin and PluginInfo objects
   OBJ_DESTROY, [self.parents, self.children, self.plugin, $
     self.plugin_info]
END


;+
; Set individual property values.
;
; @keyword name {in}{optional}{type=string} Name of the product.
; @keyword description {in}{optional}{type=string} Description of the
;                    products.
;-
PRO Product::SetProperty, NAME=name, DESCRIPTION=description
   IF N_ELEMENTS(name) THEN self.name = name
   IF N_ELEMENTS(description) THEN self.description = description
END
```

```
;+
; Get individual properties.
; <i>(See SetProperty methos)</i>
;-
PRO Product::GetProperty, NAME=name, DESCRIPTION=description
   IF ARG_PRESENT(name) THEN name = self.name
   IF ARG_PRESENT(description) THEN description = self.description
END



;+
; Sets a parent dependency for the object and automatically sets a
; reference from parent to child.
;
; @param product {in}{required}{type=Product} Product object that
;               represents a parent dependency.
;-
PRO Product::AddParent, product
   IF NOT OBJ_ISA(product, 'Product') THEN BEGIN
      logmsg, 'ERROR', 'The object is not a Product object !!!'
      RETURN
   ENDIF
   self.parents->Add, product
END



;+
; Sets a child dependency for the object and set a reference from
; child to parent.
;
; @param product {in}{required}{type=Product} Product object that
;               represents a child dependency.
;-
PRO Product::AddChild, product
   IF NOT OBJ_ISA(product, 'Product') THEN BEGIN
      logmsg, 'ERROR', 'The object is not a PRODUCT object !!!'
      RETURN
   ENDIF
   self.children->Add, product
END



;+
; Gets the parent dependencies of the product
;
; @returns Linkedlist of Product.
;-
FUNCTION Product::GetParents
```

```
   RETURN, self.parents
END


;+
; Gets the children referencies.
;
; @returns Linkedlist of Products
;-
FUNCTION Product::GetChildren
   RETURN, self.children
END


;+
; Executes the plugin associated with the Product to generate the
; output data.
;-
PRO Product::MakeProduct

   ;;Test if it was executed yet
   IF self.executed THEN RETURN

   ;;Test if all parents are executed before.
   num = self.parents->Get_Count()
   executed = 1
   i = 0
   WHILE i LT num AND executed EQ 1 DO BEGIN
      product = self.parents->Get_Item(i, /DEREFERENCE)
      executed = product.executed
      i++
   ENDWHILE
   IF executed NE 1 THEN RETURN

   ;;Collect the input parameters for the Product.
   input_params = self.plugin_info->CollectInputData()
   self.plugin->SetInputData, input_params

   ;;The node is not executed yet. We execute it and all its children
   logmsg, 'DEBUG', 'Go to execute the plugin of the product: ' + self.name
   result = self.plugin->Execute()
   logmsg, 'DEBUG', 'Result was: ' + STRING(result)

   ;;Test if there was errors.
   ;;If all is ok then mars as executed, otherwise returns and
   ;;don't execute its children.
   IF result EQ 1 THEN self.executed = 1 $
   ELSE RETURN
```

```
   num = self.children->Get_Count()
   FOR i=0, num-1 DO BEGIN
      product = self.children->Get_Item(i, /DEREFERENCE)
      product->MakeProduct
   ENDFOR
END


;+
; Sets the Plugin object on which is based the Product.
;
; @param plugin {in}{required}{type=Plugin} Plugin on which is
;              related.
;-
PRO Product::SetPlugin, plugin
   IF NOT OBJ_ISA(plugin, 'Plugin') THEN BEGIN
      logmsg, 'ERROR', 'The object is not a PLUGIN object !!!'
      RETURN
   ENDIF
   self.plugin = plugin
END


;+
; Sets the PluginInfo object that stores the Plugin information.
;
; @param plugin_info {in}{required}{type=PluginInfo} PluginInfo object.
;-
PRO Product::SetPluginInfo, plugin_info
   IF NOT OBJ_ISA(plugin_info, 'PluginInfo') THEN BEGIN
      logmsg, 'ERROR', 'The object is not a PLUGIN_INFO object !!!'
      RETURN
   ENDIF
   self.plugin_info = plugin_info
END


;+
; Creates an association from PluginInfo to Plugin object of the
; Product.
;
; @pre Product object must has references to a valid Plugin and
; PluginInfo objects.
;-
PRO Product::AssocPluginInfo2Plugin
   IF OBJ_ISA(self.plugin, 'Plugin') AND $
     OBJ_ISA(self.plugin_info, 'PluginInfo') THEN BEGIN
```

```
      self.plugin_info->SetPlugin, self.plugin
   ENDIF
END


;+
; Gets the Plugin object on which the Product is based.
;
; @returns Plugin
;-
FUNCTION Product::GetPlugin
   RETURN, self.plugin
END


;+
; Gets the PluginInfo object with the information of the Plugin on
; which the Product is based.
;
; @returns PluginInfo
;-
FUNCTION Product::GetPluginInfo
   RETURN, self.plugin_info
END
```

## File Attachments

1) product__define.pro, downloaded 100 times