

Módulos

Bienvenidos

Mi nombre es Silvia Keymetlian y trabajo en el equipo de Soporte de GeneXus.

Hoy les voy hablar del objeto Módulo, un nuevo objeto en GeneXus X Ev3. Les voy a mostrar cómo utilizarlo y las ventajas que brinda este objeto para el entendimiento, mantenimiento e integración de una KB.



Working With Modules

Modules vs. Folders

KB Conversion

Support

Vamos a comenzar la presentación sobre módulos y la vamos a dividir en 4 temas.

Primero explicaremos que es el objeto módulo y como se utiliza desde GeneXus tanto en base de conocimientos que ya existen como en nuevas bases de conocimiento.

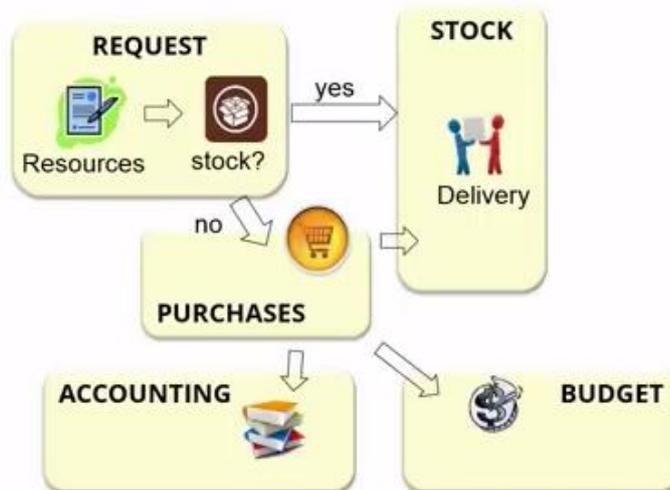
Luego vamos a ver el comportamiento de los módulos con respecto a los folders.

Después que sucede cuando convertimos una KB a una nueva versión y los aspectos a tener en cuenta sobre la compatibilidad.

Y por último que generadores soportan esta funcionalidad y cuáles no.

Comenzamos entonces viendo como trabajar con módulos desde GeneXus.

What?



Vamos a comenzar viendo un ejemplo de un ERP, como el que le muestro en la figura, donde se solicita un recurso y si se tiene stock, ese recurso si está disponible se entrega y si no se debe realizar una orden de compra para comprar ese producto o ese recurso que se necesita.

Para eso entonces se tiene que ver si tiene presupuesto, en caso que se tenga presupuesto se realiza la compra y una vez realizada la compra se debe contabilizar.

Luego de comprado el recurso se entrega como se hizo anteriormente.

Entonces en este caso, cada uno de estos iconos que estamos mostrando aquí incluye varios objetos. Esos objetos generalmente se agrupan para mejor entendimiento de la aplicación y para que por ejemplo cada desarrollador se concentre en un grupo de objetos o en la funcionalidad que está desarrollando.

En ese caso los módulos van ayudar en el sentido de que nosotros podemos definir un módulo en GeneXus y agrupar esos objetos en un módulo GeneXus.

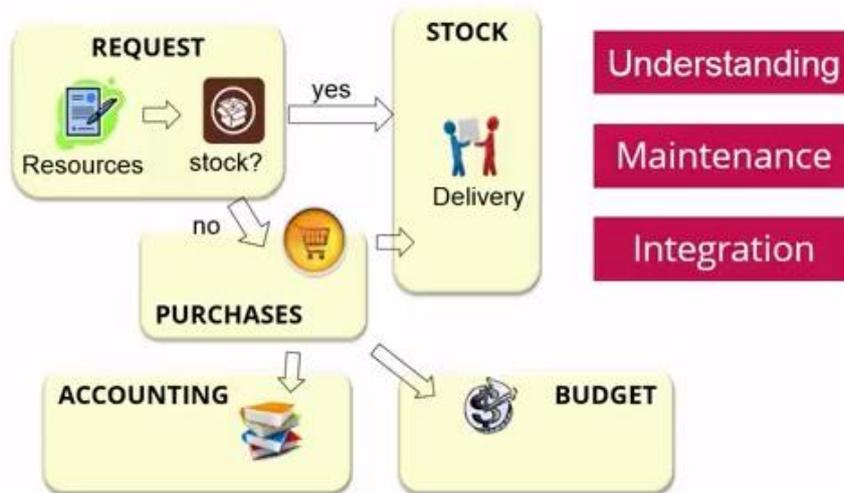
También nos va a beneficiar el uso de módulos al entendimiento de la KB, al mantenimiento de la KB y a poder integrar objetos de un módulo con otro módulo.

Incluso si un desarrollador nuevo ingresa en el equipo, es mucho más fácil contarle al desarrollador cierta funcionalidad mostrándole los objetos de un módulo que si no estuvieran ordenados u organizados dentro de los módulos.

Esto antes con GeneXus lo podíamos hacer con el uso de folders, con el uso de nomenclatura especial para indicar que objetos pertenece a cierta funcionalidad pero ahora tenemos una formula mucha más fácil de hacer.

Por eso decimos que el uso de módulos nos permite entender mejor la KB, el mantenimiento de la KB y la integración.

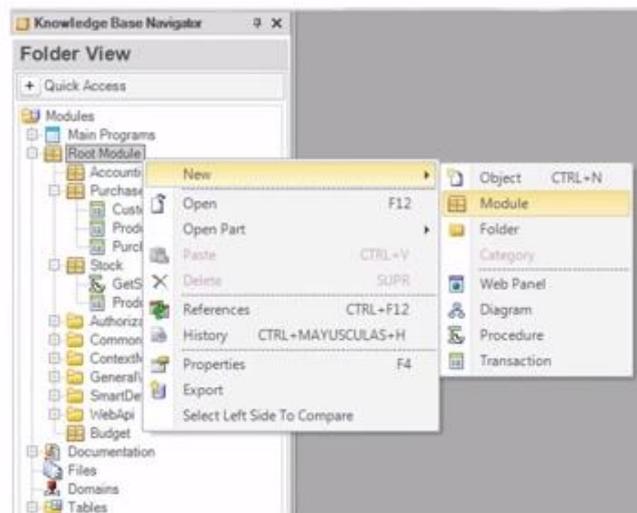
What?



¿Cómo hacemos para definir un módulo en GeneXus?

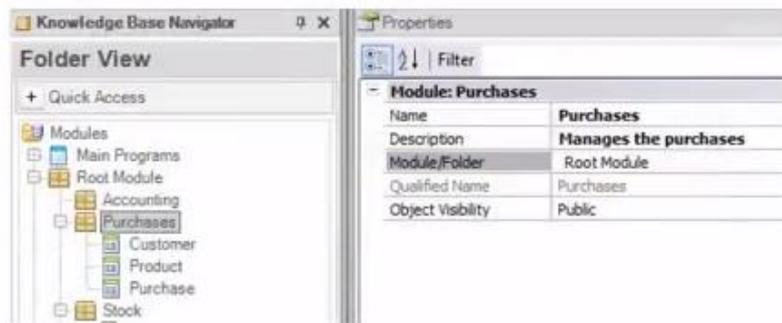
Simplemente dando botón derecho sobre el Root Module o sobre otro módulo que ya se tenga, y como cualquier otro nuevo objeto “new / module”

How?



De esa forma definimos un nuevo módulo.

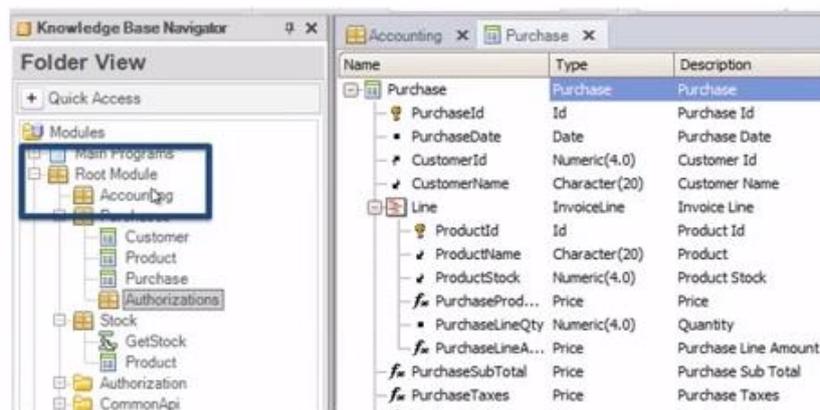
How?



Una vez que lo vamos a definir se solicitan ciertas propiedades como son el nombre, la descripción y algunas otras propiedades asociadas a los módulos.

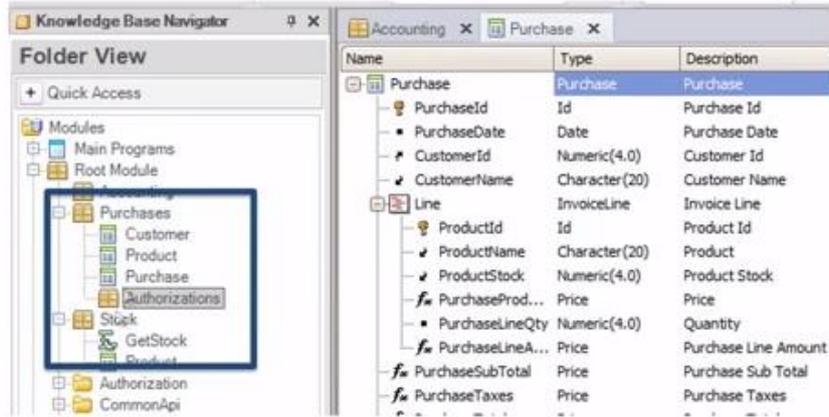
Una vez que se define el módulo queda una estructura similar a lo que es una estructura de folders. Y en la versión Evolution 3 todos los objetos van a pertenecer algún módulo. Si nosotros no definimos módulos propios por defecto siempre se crea el root module que se crea para todas las KB en la Evolution 3.

How?



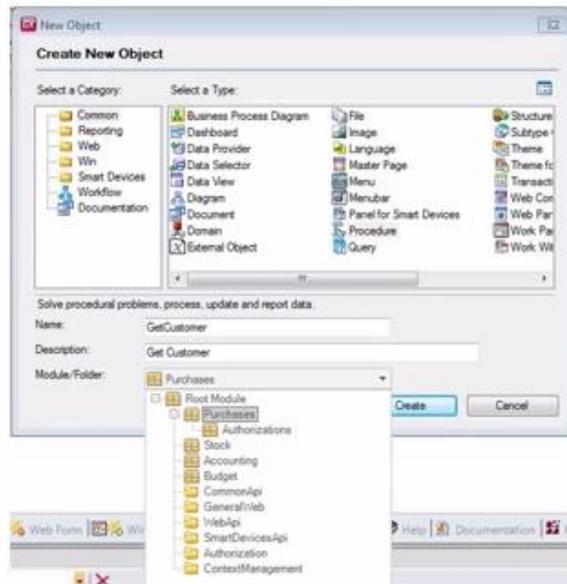
Y además un módulo puede tener su módulo en este caso el módulo purchase que tiene un sub módulo authorizations.

How?



También cada vez que definimos un objeto podemos indicar a que módulo pertenece, desde la opción module folder podemos indicar ese objeto a que módulo va a pertenecer.

New Object



Y además cuando definimos nuevos módulos con drag and drop podemos incluir objetos dentro de ese módulo recién creado.

¿Cuáles son los objetos que se pueden definir dentro de un módulo?

Los dominios, las imágenes, el lenguaje, los temas, los atributos, las tablas, los archivos: no van a pertenecer a ningún módulo ya que son objetos globales a toda la base de conocimiento.

Los objetos Win como los work panels, menú y menú bar solamente pueden pertenecer al Root Module ya que ya que los generadores Win no soportan esta funcionalidad.

El resto de los objetos que aparecen en la columna de más abajo...

Which Objects can be Defined in a Module?

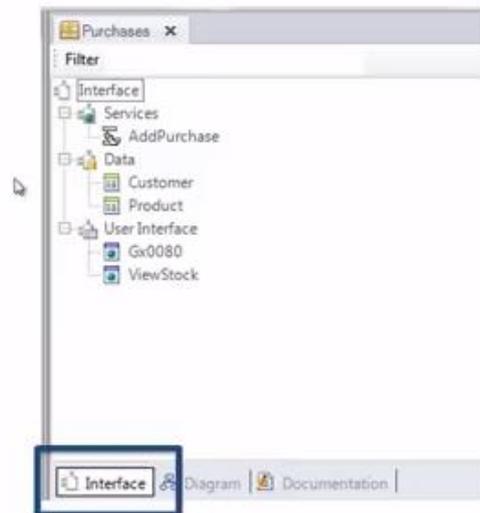
Object	Module
Domain, Image, Language, Theme, Attribute, Table, File	No
Work Panel, Menu, Menu Bar	Root Module
Folder, Transaction, Procedure, Web Panel, Panel for SD, Data Selector, SDT, Diagram, Document, External Object, Subtype Group	Yes

..Sí pueden pertenecer a cualquier módulo que nosotros definamos en la KB.

¿Cómo está compuesto un módulo?

Nosotros una vez que lo definimos podemos abrirlo haciendo botón derecho “open” sobre el módulo, y ahí vamos a ver que hay tres tabs:

Interface Tab



Uno llamado Interface, el otro llamado Diagram y el otro Documentation

Dentro del tab interface, ese está dividido en tres secciones. Uno que se llama Services, otra Data y la otra User Interface.

En la sección Services se define lo que llamamos la Api del módulo, o sea los objetos incluidos en esta sección pueden ser Procedimiento, Data Provider y Extended Object

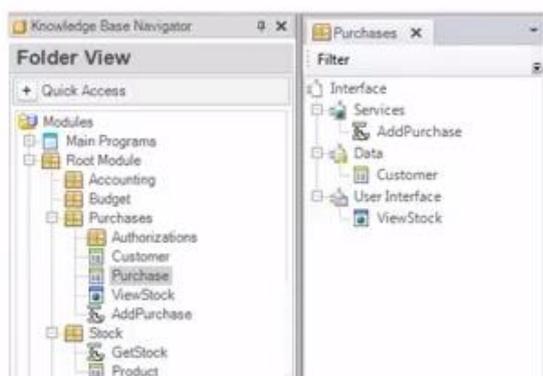
Dentro de la sección Data se incluye las transacciones, los business component y los SDT. Y dentro de la sección User Interface se incluyen las master page, los web components y los web panels.

Cada vez que un objeto se incluye dentro de un módulo automático se organiza en alguna de estas tres secciones.

Vamos a ver entonces como se determina que objetos aparecen en el tab interface y cuáles no.

En la figura vemos el módulo Purchase que veíamos antes...

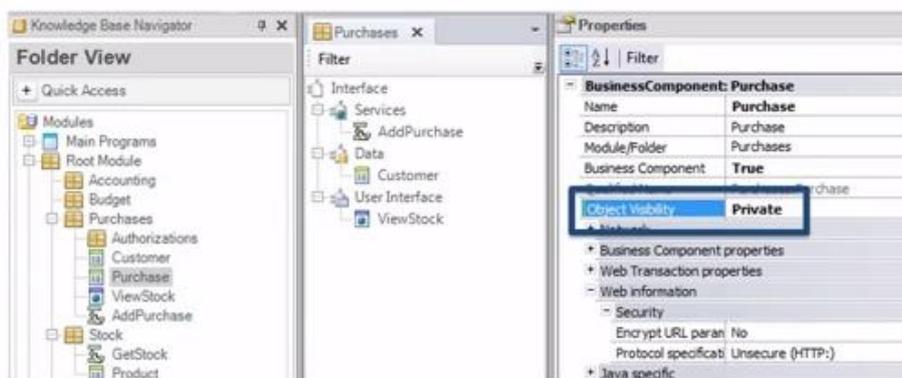
Object Visibility Property



...acá la diferencia está en que cuando se abre el módulo en el tab interface no se ve la transacción purchase y sí se ve la transacción customer

¿Por qué es esto? Esto es porque la transacción Purchase tiene la propiedad Object Visibility en al valor Private. Esto es una nueva propiedad de los objetos a partir de la introducción de los módulos en la KB.

Object Visibility Property



Los objetos que pertenecen a un módulo tienen una propiedad Object Visibility con valores public y private.

Public significa que el objeto puede ser utilizado de cualquier punto de la base de conocimiento y son los que se ven en el tab interface.

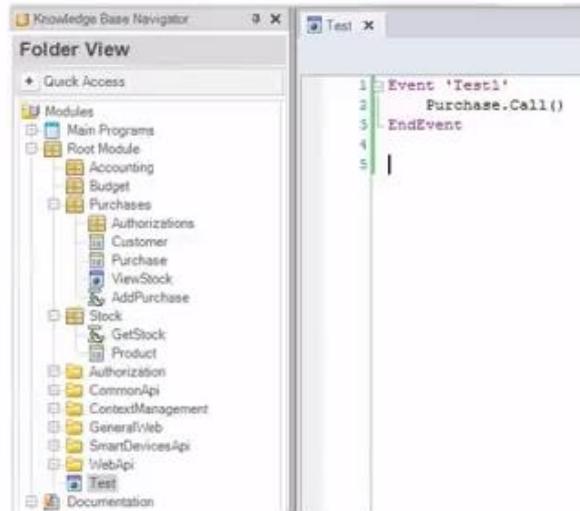
Y Private significa que puede ser usado solo dentro del módulo, esto incluye a los submódulos también. O sea un submódulo puede usar los objetos privados del módulo padre.

Por defecto, todos los objetos tienen la propiedad public y la lista de esos objetos públicos son los que conforman la interfaz del módulo. Los objetos privados no se ven en el tab interface, esto significa que no se puede usar este objeto desde ningún otro objeto de la KB.

Por ejemplo, si se quiere hacer un call de la TRN Purchase desde un objeto del Root module como este caso...



Object Visibility Property



...va a dar un error de especificación.



Object Visibility Property



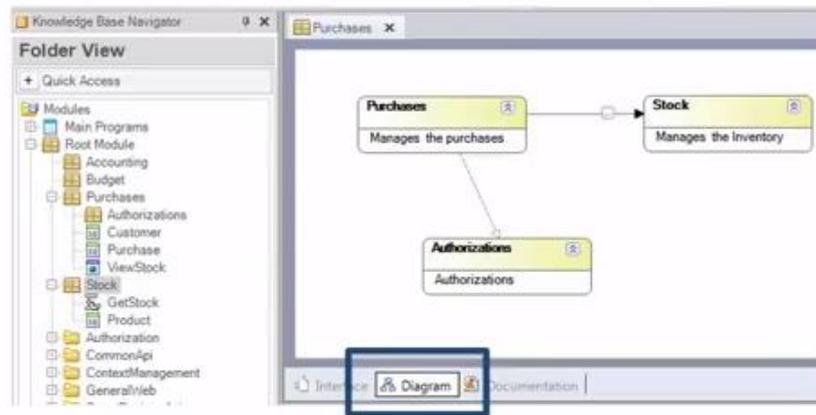
Eso es porque ese objeto está como privado y no puede ser llamado desde otro objeto.

Es ahí entonces donde se deben definir las interfaces para poder acceder a ese objeto pero no se va a poder hacer un call directo a ese objeto. De esa forma se maneja cuáles son los objetos en el tab interfaces y cuáles no.



El otro tab que aparece dentro el objeto módulo es el tab Diagram.

Diagram Tab



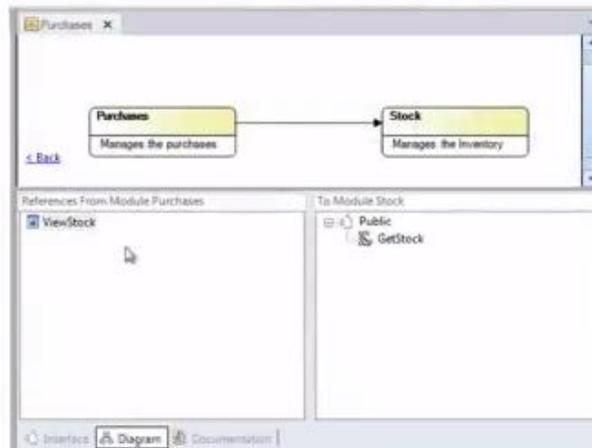
Ese tab lo que nos permite ver es como se relaciona un módulo con el resto que puede ser referenciado o puede ser un submódulo.

La relación de submódulo se da cuando un módulo es parte de otro módulo; como en este caso Authorization que es parte del módulo Purchase es un submódulo de purchase. Es ahí cuando la relación se muestra con esta línea punteada.

En cambio la relación de referencia entre módulos se da con una línea completa y en este caso indica que desde el módulo de purchase se utiliza un procedimiento que se encuentra dentro del módulo stock.

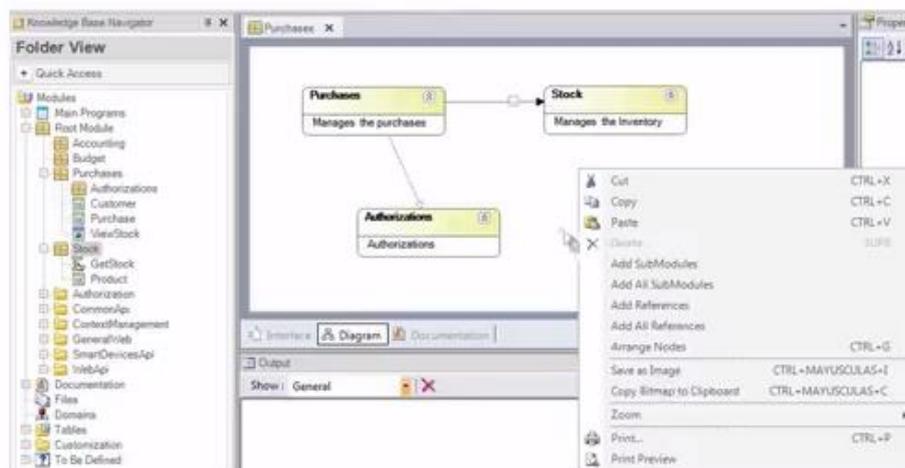
Incluso lo que podemos hacer es haciendo doble click ver del detalle que objetos pertenecen a un módulo y al otro.

Diagram Tab



Además dando botón derecho sobre el tab diagram hay un menú contextual con otras opciones que nos permiten por ejemplo agregar con otros submódulos o agregar todos los submódulos para ver como se relacionan entre ellos.

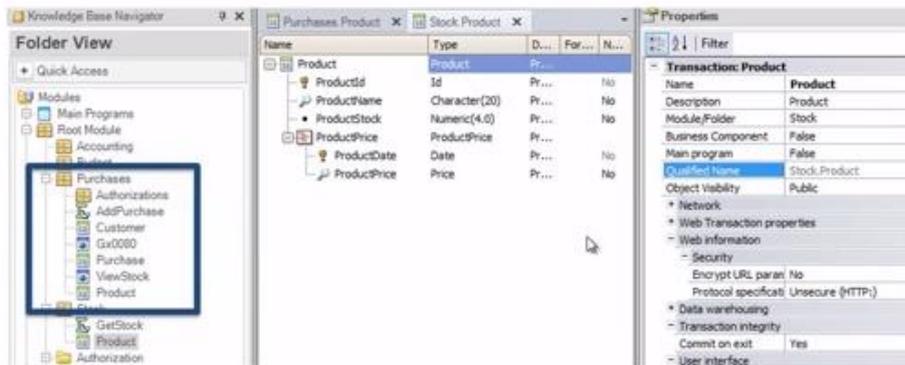
Diagram Tab



Otro aspecto a tener en cuenta cuando se trabajan con módulos es que se puede tener objetos con el mismo nombre con diferentes módulos.

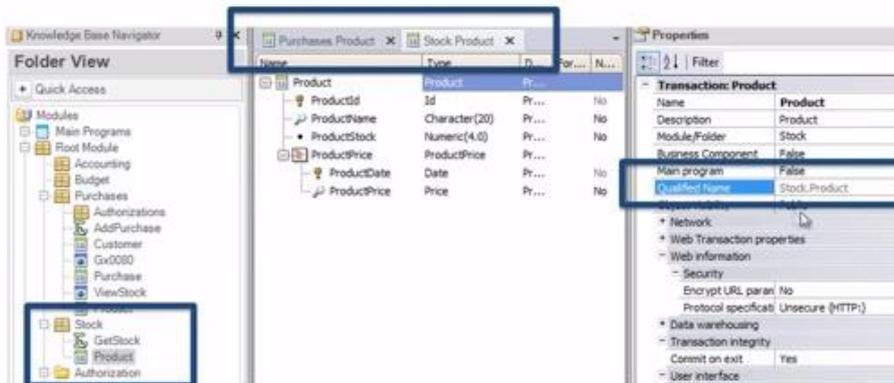
En este caso tenemos el objeto product dentro del módulo Purchase y tenemos el objeto product dentro del módulo Stock.

Qualified Name Property



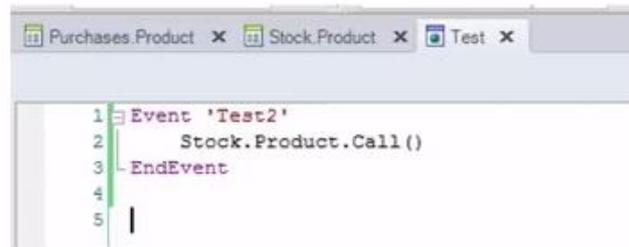
Entonces lo que podemos hacer para determinar exactamente de qué objeto estamos hablando es indicarlo con la propiedad QualifiedName que es el nombre del módulo punto (.) el nombre del objeto.

Qualified Name Property



De esta forma podemos tener dos objetos con el mismo nombre en diferentes módulos. Y cuando queramos referenciar a los objetos vamos a tener que también escribirlos; Nombre del módulo punto (.) nombre del objeto...

Grammar



```
Purchases.Product x Stock.Product x Test x
1 Event 'Test2'
2     Stock.Product.Call ()
3 -EndEvent
4
5 |
```

...salvo para los objetos que se encuentren en el Root module ya que no pertenecen a ningún módulo. Y esto es solamente necesario cuando tenemos ambigüedad, cuando tenemos dos objetos con el mismo nombre y si el objeto es único dentro de la KB no es necesario referenciarlo con el nombre del módulo antes.

También hay que tener en cuenta que cuando se usan módulos en la KB cambia la URL de los objetos de nuestra aplicación. Esto es porque cuando ejecutamos un objeto, a partir de ahora va a aparecer el nombre del módulo antes del nombre del objeto para cualquier generador.

URL Syntax

Java

.../servlet/packageName.module.object

.NET

.../module.object.aspx

Ruby

.../module.object.html

SOAP Services

.../module.object?wsdl

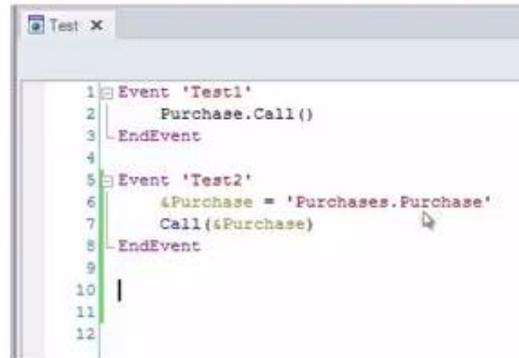
Rest Services

.../rest/module/object/1

Y específicamente para el generador JAVA haya que usar obligatoriamente la propiedad packagename para indicar el nombre del package y también se va a adicionar a la URL que se genera para referenciar al objeto.

Como vimos anteriormente la transacción purchase tiene la propiedad object visibility en private por lo tanto no podemos hacer un call directo a esa transacción pero si es posible hacer un call dinámico a esa transacción.

Dynamic Calls



```
1 Event 'Test1'
2     Purchase.Call()
3 EndEvent
4
5 Event 'Test2'
6     &Purchase = 'Purchases.Purchase'
7     Call(&Purchase)
8 EndEvent
9
10 |
11
12
```

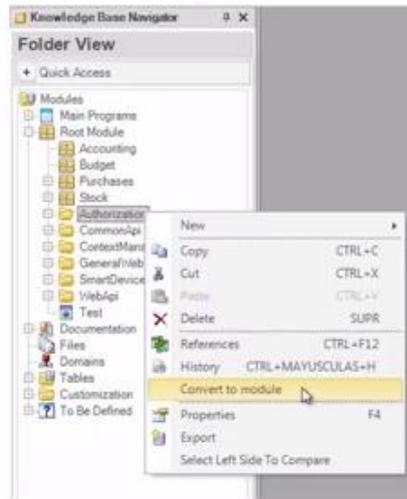
Por ejemplo en este caso, en el evento test2 estamos invocando a la transacción purchase como un call dinámico. El único tema a tener en cuenta en este caso es que si se tiene la propiedad expand dynamic call en Yes; si esta transacción tiene la propiedad object visibility private nunca será accesible por su llamador, por lo tanto no se incluirá dentro de los objetos posibles a ser ejecutado.

Veamos ahora la diferencia entre usar un módulo y un folder dentro de una KB GeneXus.

Modules vs. Folders

Por ejemplo una cosa que podemos hacer es que un folder se puede convertir en un módulo - esto se hace haciendo botón derecho sobre un folder - y convertirlo en un módulo. Esto sería una forma fácil de pasar los folder que nosotros veníamos usando a que sean módulos.

Convert a Folder into a Module



Entonces como vimos tanto los módulos como los folder nos ayudan a organizar los objetos en una KB, juntos crean un árbol jerárquico donde la raíz es el Root Module y en el diálogo de folder view podemos ver ese árbol que se va formando pero sin embargo hay diferencias conceptuales entre un módulo y un folder.

	Module	Folder
Used to organize objects in a KB	✓	✓

Una de ellas es que los módulos son parte de la propiedad Qualified Name pero los folder no. Eso es lo que nos permite poder tener dos objetos con el mismo nombre en diferentes módulos.

	Module	Folder
Used to organize objects in a KB	✓	✓
Qualified Name property	✓	✗

Además los módulos pueden tener folder como hijos pero los folders no pueden tener módulos como hijos.

	Module	Folder
Used to organize objects in a KB	✓	✓
Qualified Name property	✓	✗
Can have child Folder/Module	✓	✗

¿Cómo hacer entonces para decidir cuándo usar un folder o un módulo?

Como regla general lo que podemos decir es que se puede usar los módulos para encapsular y los folders para organizar los objetos dentro del módulo.

	Module	Folder
Used to organize objects in a KB	✓	✓
Qualified Name property	✓	✗
Can have child Folder/Module	✓	✗
Use for	Encapsulation	Organization within Modules

¿Qué tenemos que tener en cuenta cuando convertimos una KB de versiones anteriores a la Ev3 con respecto a los módulos?

KB Conversion

- Changes to the KB
 - Root Module is created
 - All objects are assumed to belong to the Root Module
 - Dots (.) are replaced for Underscore (_) in the Name property
- Code Generation: Not affected

Es que por defecto siempre se crea el Root Module, además todos los objetos se asumen que pertenecen a ese Root Module salvo que definamos nuevos módulos y esos objetos los

movamos a los nuevos módulos y si tenemos objetos que tienen puntos (.) dentro del nombre se van a reemplazar por un underscore (_) en la propiedad name.

En cuanto al código generado si nosotros no definimos nuevos módulos en la KB, el código generado no cambia solamente va afectar cuando empezamos a introducir módulos en la KB. Y con respecto a la compatibilidad, una vez que se abre la KB con la Ev3 no se puede abrir con versiones anteriores, no hay compatibilidad hacia atrás.

Y en el caso de usar GXserver también se debe tener GXserver en la Ev3. Se debe usar GeneXus y GXserver en la misma versión en la Ev3.



KB Conversion

- **Changes to the KB**
 - Root Module is created
 - All objects are assumed to belong to the Root Module
 - Dots (.) are replaced for Underscore (_) in the Name property
- Code Generation: Not affected
- **Compatibility**
 - No backward compatibility
 - GX and GXserver Ev3

Como vimos entonces en cuanto al soporte de esta funcionalidad los generadores que soportan esta funcionalidad son los generadores Java, Net, Ruby y el generador para Smart Devices. Los generadores WIN no están soportando esta funcionalidad.



Supporting Modules

- Java Web
- .NET Web
- Ruby
- Smart Devices

Not Supporting Modules

- Java Win
- .NET Win
- Cobol
- RPG
- Visual FoxPro
- .NET Mobile

Lo que me resta es invitarlos a probar GeneXus XEv3 y comprobar los beneficios de esta nueva funcionalidad, por cualquier duda les dejo mi mail silvia@genexus.com

Thank you!

More Info & documentation
wiki.genexus.com

Silvia Keymetlian
silvia@genexus.com