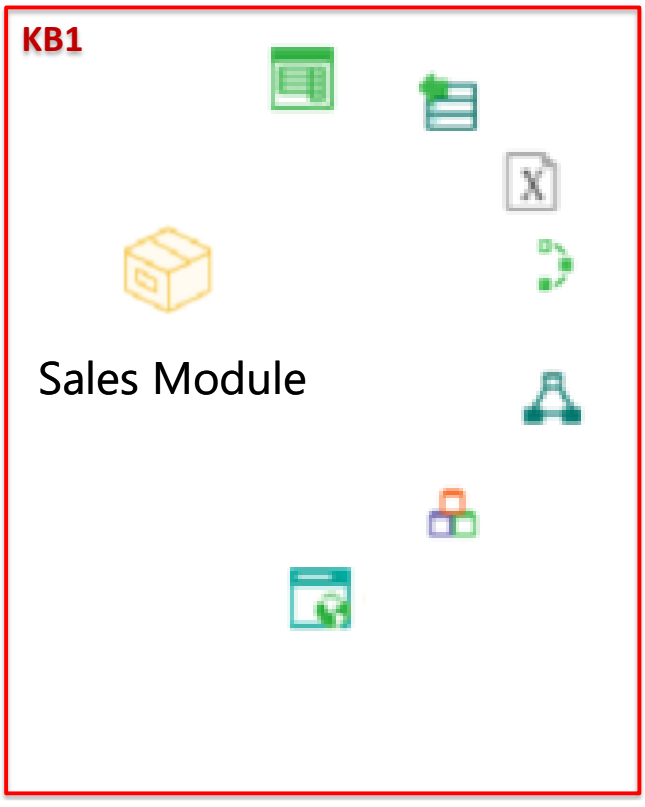
Modules and External Objects

GeneXus

Here, we will consider some GeneXus objects that prove useful at the time of encapsulating functionalities and organizing objects in our knowledge base.

Modules

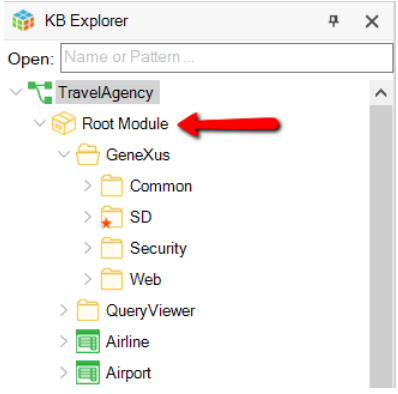
Let’s consider modules to start with.

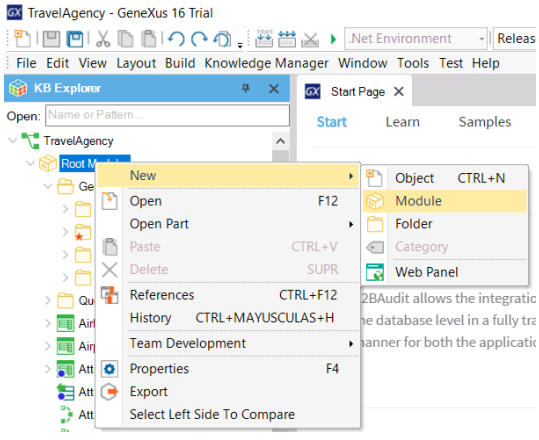


Modules are GeneXus objects acting as containers that allow us to group objects from our KB, making it easier to understand and to maintain the KB, as well as to integrate objects with other KBs.

What is a module?

- GeneXus object that enables us to group objects from the KB and encapsulate its functionalities.
- Designed to facilitate the understanding, maintenance and integration of objects among KBs.
- Modules and folders enable us to create hierarchies.





When we create a knowledge base, the Root Module is then created and, by default, all the objects we create remain in that module.

Modules and folders aid us in organizing objects. However there are conceptual differences between modules and folders. Modules help in encapsulating and modularizing parts of the KB, with the possibility of determining which objects are visible from other objects and which are not, as we will see later.

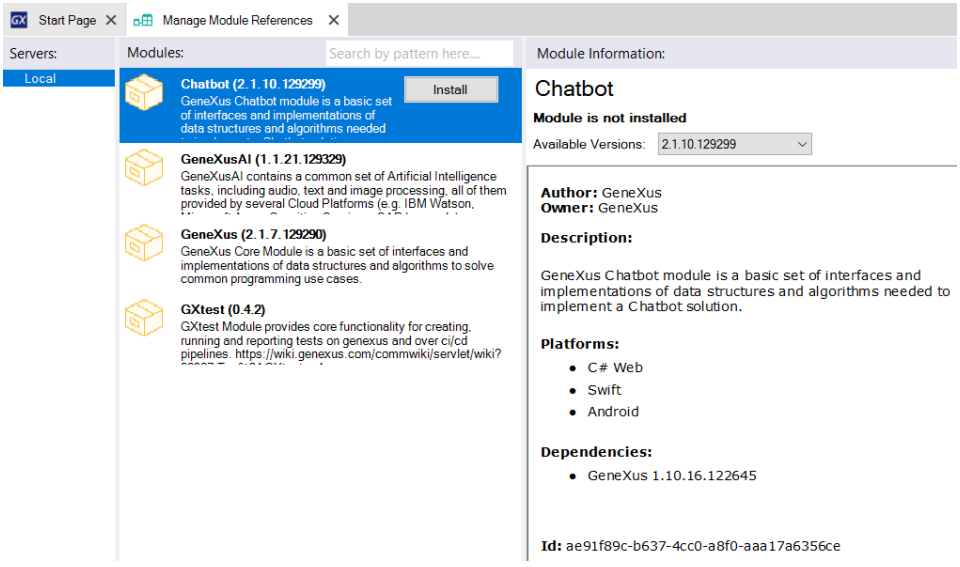
Folders, on the other hand, act as containers that only help in organizing objects by separating them according to specific criteria. Along with modules, they create a hierarchical tree where the root is always a module as in the case of the Root Module. We can see this in KB Explorer.

Modules may have module children, but folders may not have modules as their children.
As a general rule, we could say that it is possible to use modules to encapsulate and folders for organizing objects within the module.

In order to add an object to a module, we could drag it to the module in KB Explorer, or otherwise click the right button on the module and then New Object, or otherwise change the value of the object's Module/Folder property.

Adding already created modules to the KB

- Knowledge Manager / Manage Module References



Packed modules that have been shared with us may be viewed through the Knowledge Manager / Manage Module References menu.

For each module available, we can see its information to decide whether we will install it in our KB or not. If we do, it will be saved under the References node of KB Explorer, as opposed to objects we create, which are saved in the Root Module by default.

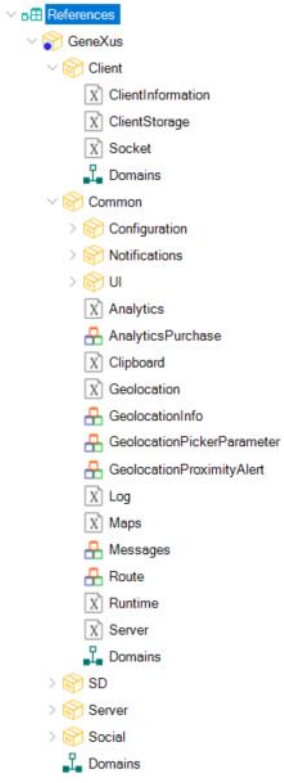
We cannot modify the objects of these modules (we view them as Read-Only) which are already compiled. So, when we press F5, it is not necessary to specify or generate them and so on.

However, we will be able to use them freely from the objects in our application, using all the functionalities they have available.

Any member of the community may create, share or even sell his/her modules through Marketplace.

One of these is the GeneXus module, also known as GeneXus Core.

GeneXus Module



The GeneXus module is distributed automatically and installed in all KBs. And as any external mode of our application, we will find it in the References node.

It comprises a series of sub-modules that contain a set of APIs with their corresponding domains and SDTs, which enable us to interact with the various technologies, devices, sensors, applications and so on.

These APIs are implemented as External Objects, which we will see next.

More information on modules

https://wiki.genexus.com/commwiki/servlet/wiki?22411

You will find further information on the module object at this wiki link: https://wiki.genexus.com/commwiki/servlet/wiki?22411

External Objects

Let’s now see what External Objects are.

External objects are GeneXus objects that enable us access to external resources of our KB as if they were another object in the KB.

That’s why they are increasingly more frequent and important in our web and mobile device applications. Let’s now see how to use them.

What resources do we have as external objects for our KB?

- Native objects created and compiled in a programming language:
  - Assemblies of .NET (.dll)
  - Java classes (.class)
- Resources stored in different external sources:
  - Enterprise Java Beans (EJB), of a JEE server
  - Stored Procedures created on a DBMS
  - Web Services (WSDL, OpenApi) published on a web server
  - SAP BAPI Modules
  - JSON Files
  - XML Schema
- External Objects available in GeneXus
  - APIs for Smart Devices: access to HW (camera, GPS, microphone, etc.), and SW (calendar, contacts, notifications, etc.)
  - APIs to access events, communications with the server, notepad, maps, social networks, etc.
- External Objects published on the GeneXus Marketplace

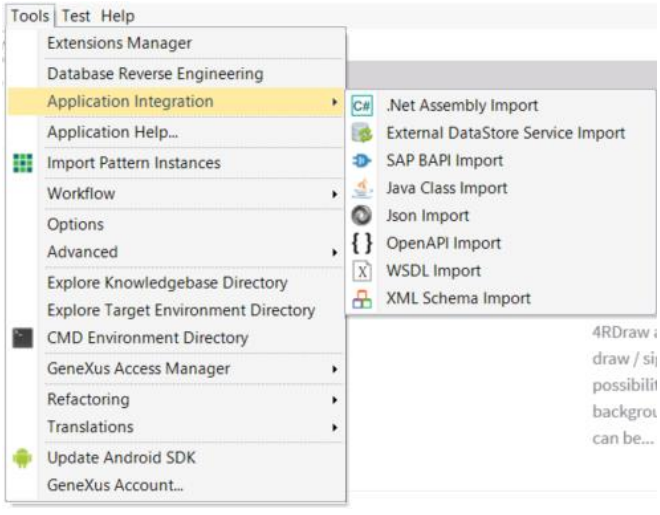
We may import different types of resources into our KB. For instance, when we have something programmed in .NET we may generate a DLL and import it into the KB as an external object. Then, from our app, we may invoke the functions included in the DLL as if they were procedures programmed in GeneXus.
The same happens with classes created in Java.

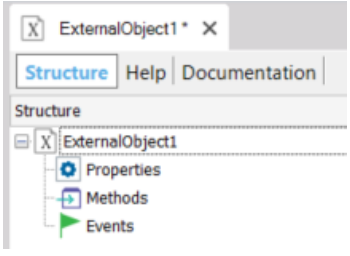
We may also import resources stores in other external sources, like Java Beans programs, procedures stored in a database, webservices (both SOAP and REST), SAP modules, JSON files generated by any application, or XML schema.

GeneXus also provides a set of External Objects that are located in the RootModule or in the GeneXus module that enable us access to a variety of resources such as APIs for interacting with the hardware, or native applications of mobile devices, or APIs to have access to the server, to events, or to Windows applications such as the notepad, as well as to external sites for using maps and social networks, among other things.

There are also external objects published on the GeneXus Marketplace that we may include in our application.

Create an external object using the Wizard



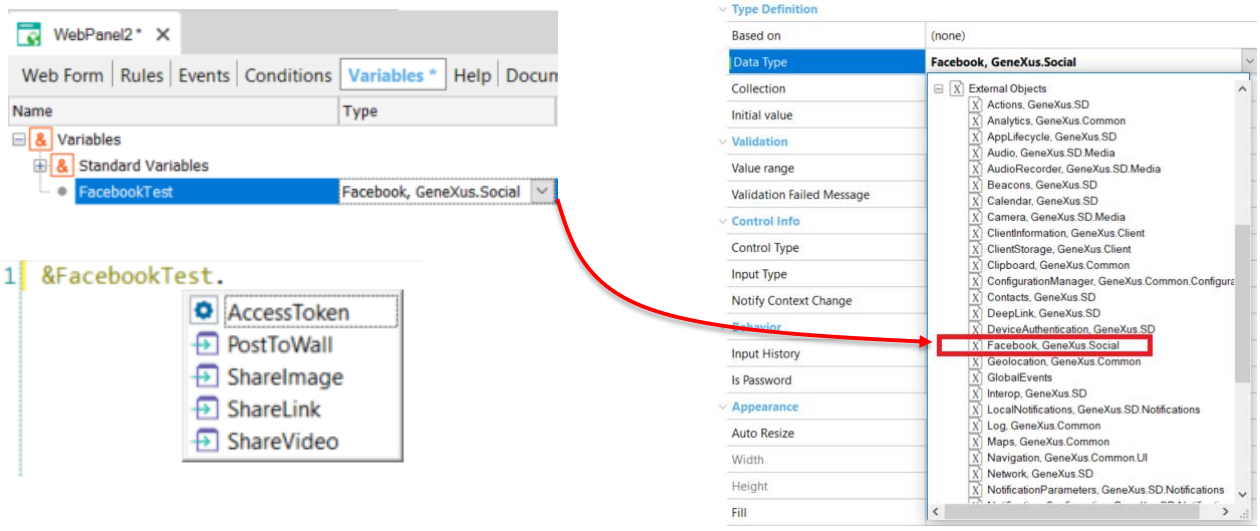


The best way to create an external object is to use a wizard. If we go to the Tools menu and select Application Integration, we will see the various resources to be imported, and a specific wizard will be executed for that resource. Upon finalizing the wizard, the external object created will be automatically associated with the resource. All the properties of the external object will be adjusted in accordance with the type or resources that has been imported.

We may also create an external object with New Object, just like with any other GeneXus object, though in that case, we will need to set up its properties, methods and events manually.

How are external objects used?

- 1) We create a variable with the External Object type
- 2) We invoke the methods or assign the properties available

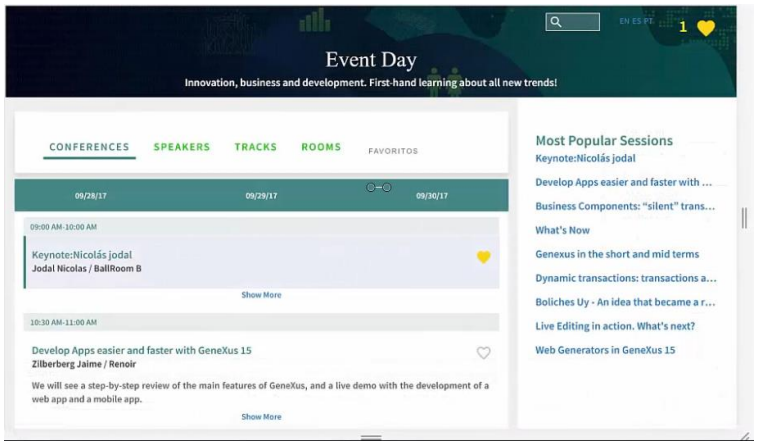


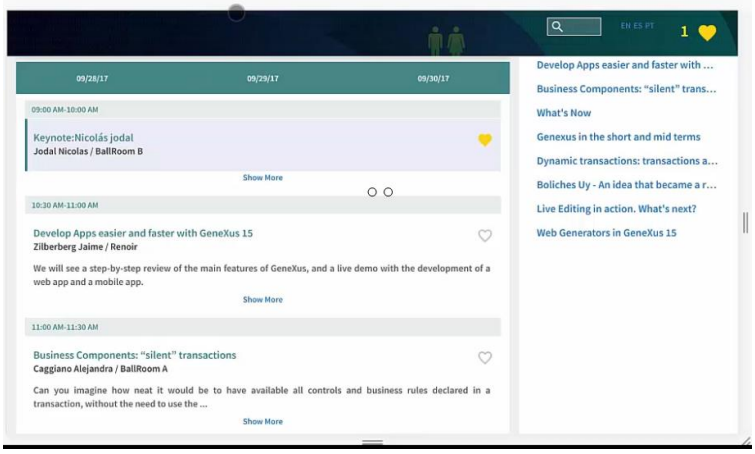
Once we have created the external object based on the properties that correspond to the external object that we wish to use, it will be available just like any other data type in the knowledge base.

We use it in the same way as any other type of extended data, by defining a variable of that type and then calling the methods and/or setting up the properties we need.

You will find further information on external objects if you go to this link on screen: https://wiki.genexus.com/commwiki/servlet/wiki?5669

Example of use of an external object with JavaScript





Another thing we can do with External Objects is interact with JavaScript; for example, to link events implemented in an external JavaScript to a GeneXus event.

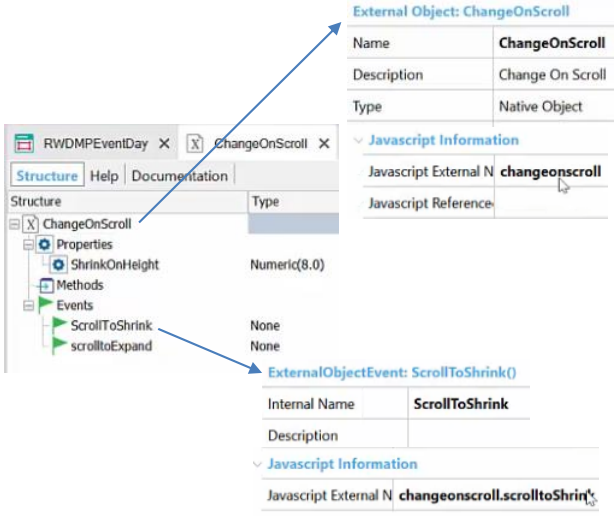
Let's see this with a sample application.

Note that when we scroll in the application, the top bar gets smaller.
This is implemented with a JavaScriptChangeOnScroll event programmed externally. Let's see how to link this JavaScript to GeneXus.

Example of use of an external object with JavaScript (continued)

changeonscroll.js

```
var changeonscroll = {
    shrinkOnHeight: 30
};
$(function() {
    $(window).on('scroll', function() {
    var distanceY = window.pageYOffset || document.documentElement.scrollTop;
    var shrinkOn = changeonscroll.shrinkOnHeight;
    if (distanceY > shrinkOn ) {
        gx.fx.obs.notify("changeonscroll.scrolltoShrink");
        }
     else {
        gx.fx.obs.notify("changeonscroll.scrolltoExpand");
        }
    });
});
```




```
Event Start

    Form.HeaderRawHTML = !"<link href='https://fonts.googleapis.com/css?family=Source+S

    Form.HeaderRawHTML += GetChangeOnScrollScript()
    changeonscroll.ShrinkOnHeight = 20
```

The JavaScript we have is very simple. Basically, when a certain scroll height has been reached, it triggers a Shrink event; otherwise, it triggers an Expand event.

How do we link this JavaScript to GeneXus?

We do this with an external object called Changeonscroll, which is basically associated with an external JavaScript called Changeonscroll, and here are the events this JavaScript is triggering.
In this case, they have the same name as in the JavaScript but we could change it.

These events were implemented in the Web Master Panel (or master page) where the external object Changeonscroll is included with the ScrolltoExpand event and the ScrolltoShrink event.
In one case we are hiding the component and in the other case we are showing it, since basically we want it to be displayed or hidden depending on how far down the scroll bar is.

To include the JavaScript in the application, we're doing it the same way we have always done it, which is to add the script to the HTML code.

For more information about external objects:

https://wiki.genexus.com/commwiki/servlet/wiki?5669

For more information about external objects, visit the following link displayed on the screen.


GeneXus™