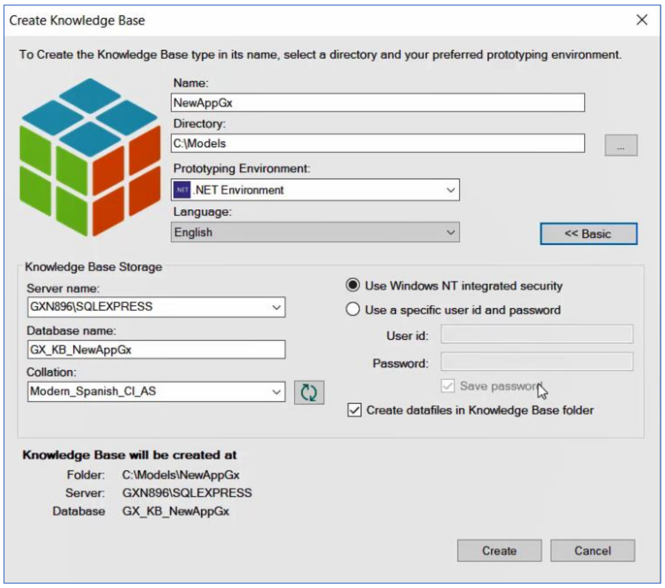
Setting some properties


Create Knowledge Base

To Create the Knowledge Base type in its name, select a directory and your preferred prototyping environment.

Name: NewAppGx

Directory: C:\Models

Prototyping Environment: .NET Environment

Language: English

<< Basic

Knowledge Base Storage

Server name: GXN896\SQLEXPRESS

Database name: GX_KB_NewAppGx

Collation: Modern_Spanish_CI_AS

[● Use Windows NT integrated security]

○ Use a specific user id and password

User id:

Password:

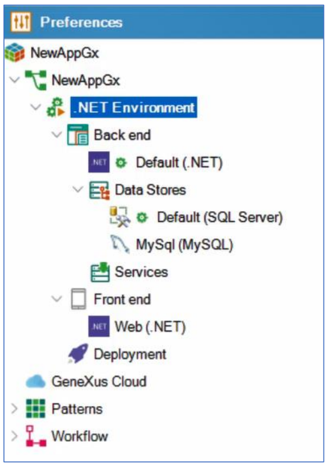
☑ Save password

☑ Create datafiles in Knowledge Base folder

Knowledge Base will be created at

Folder:	C:\Models\NewAppGx
Server:	GXN896\SQLEXPRESS
Database	GX_KB_NewAppGx

Create Cancel


Preferences

- NewAppGx
  - NewAppGx
    - .NET Environment
      - Back end
        - Default (.NET)
        - Data Stores
          - Default (SQL Server)
          - MySql (MySQL)
        - Services
      - Front end
        - Web (.NET)
      - Deployment
  - GeneXus Cloud
  - Patterns
  - Workflow

When we create a new KB, as we know, from the basic options we can configure its name, location, programming language, and language used for the labels, buttons, messages, and so on. In the advanced options, we can change the name and location of the database server where the KB database will be located, the database name, interleaving, and whether the Windows user or a specific user credentials will be used to access the database.

Once the KB is created, let's see the options displayed in the “preferences” window. Under our knowledge base version node, we see the environment automatically created from the initial definitions.
The following nodes are included:

- **Back end**, where everything related to the application server, database accesses, and services will be located.
The back end defines the programming languages that will be used to generate the code corresponding to the application's back end, through the generators.
Then the Data Stores will contain the information to access the associated database.
It is also possible to generate new Data Stores with different DBMS, for example, to obtain data from other external databases.
From the Services node, we can configure how services, storage configurations, notifications, etc. will be managed.

- Then we have the **Front end** node, where everything related to the user interface is located and connected to the back end.
The available generators to create the application Front end are shown here.
By default, the language we have selected in the environment—.Net, .Net Framework, or Java—will be displayed, and depending on the type of application being developed, the following generators may be shown: Android, Apple, and Angular.

Lastly, there is the Deployment node, which is related to the Deployment Unit objects defined in the KB Version.

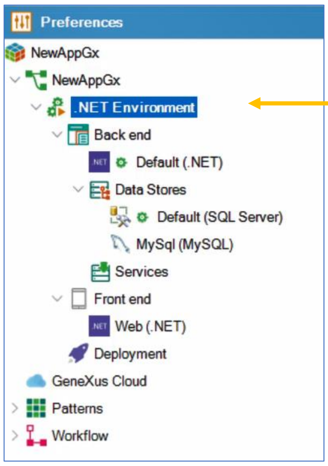
We can create new environments. For example, in developments where we have a test environment and a production environment, we need to have more than one environment.

Let's suppose we create a new one with Java generator and Oracle database, and we want it to be the default generator for our application. We can either set it to be the default environment when we create the environment, or we can do it later by right-clicking on the desired environment and selecting “Set as current Environment.” The same applies for switching environments.
To run a Java Web application, we must have previously installed the Java Development Kit (JDK), which can be downloaded from Oracle's official website.
A web server will also be necessary to run the application; in this case, we will use Apache Tomcat, an open source software that can be downloaded from its official website.

After installing it from the Windows “Services” app, we can verify that it is running correctly; otherwise, we can start it from there. Here is Apache Tomcat 9, which is the one installed on this PC, running correctly.

GeneXus

Some properties of the KB Environment node



Now let's see some configurable properties of the nodes that we have just mentioned In the KB version node, one of the properties shown is called Enable Integrated Security. This property determines whether the generated application will manage security through GAM (GeneXus Access Manager); by default it is set to false.

We also find properties that have an impact on the application interface.
For example, it is possible to set the theme or design system object to be applied by default to the different transaction and/or Web Panel objects we create. However, later on from each of these objects we can specify which theme or design system object to apply to it.

We can also define the Master Page that will be applied by default, which points to the RwdMasterPage object. As with the previous property, each Transaction and Web Panel will be able to configure this property independently.

We will also be able to configure some properties related to the validations performed on the client side, such as the stop on error property: By default it is set to No, but if it is changed to Yes, when the rules that we have defined are validated in a form and an incorrect entry is found, the focus will be kept on that field until it is corrected; it does not allow moving on to the next one without correcting it. Also, we will be able to define the position of the validation messages, among other configurations.

The Default section contains the automatic refresh property, which is set to Yes by default. As a result, when we have a grid and changes are made to any variable that is being used by the Load event, the Refresh event, or the grid conditions, the grid will be automatically refreshed without the need for the user to do it manually. The most common example of this case is when we have filters that impact the grid. If this behavior is not desired, it can be set to No.

In the KB environment node, in the Environment section we can change the platform for which the application will be generated—Web or Windows. We will also be able to modify the selected programming language from here.
In addition, we can change the assigned DBMS.

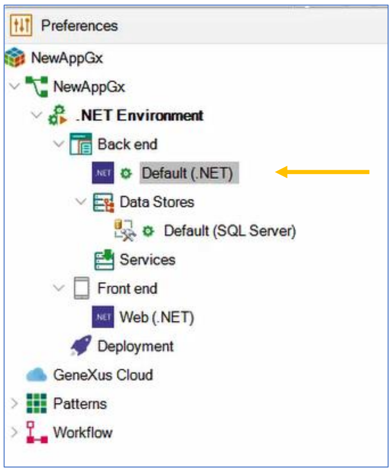
From the Startup Object property, we can assign a startup object to be executed when the application is launched; that is, when we press F5.
Also, from this property we can modify the name of the environment.
Here is the Commit on exit property; from it we set the value that it will have by default for each transaction object or procedure that we create. By default it is set to Yes, which means that the generated program executes a commit at the end of the logical unit of work (LUW). If we change it to No, this commit will not be performed.

There is also a property called Encrypt URL Parameters. With this property, we can allow or deny the encryption of the parameters sent to a URL, and set security levels when using the encryption of parameters between Web Objects. By default it is set to No, which means that the parameters in the URL of Web Objects will not be encrypted.
The session key option indicates that the URL parameters will be encrypted with a different key for each session. Encryption is done using local cookies. This value provides a higher level of security, but does not allow shared URLs. This means that a user will not be able to send a URL with parameters to another user. The reason is that the URL will not work because the corresponding cookie is required for decryption.

The last option is Site Key. If we select this the parameters in the URL of the Web objects will be encrypted, but the encryption key will be the same for the entire site. In this case, cookies are not used. This implies a lower level of security, but makes it easier to transfer links.
The protocol specification property allows specifying the protocol under which the application or Web services will be executed. By default, it will be accessible via HTTP, but we can change it to HTTPS. If we leave it unspecified, the application does not apply any protocol, so HTTP or HTTPS can be both used to access it.

GeneXus

Some properties of the back-end generator



In each generator, we also have several configurable properties and they depend on the language selected. Let's see some of .Net, for example.
We can specify the platform on which the application will be generated: web or Windows.
By means of these properties (Log Level / User Log Level), we can configure the detail of the information that will be saved in the log with a scale from 0 to 6, where 0 is the “off” option—that is, no logs will be made— and option 6—“All”—will record all the information possible.

The Reorganize server table property is used to indicate if the tables of the model are going to be reorganized or not by the generator that we have assigned as reorganizer of the model. By default it is set to Yes. If it is set to No, when the database Impact Analysis Report indicates that the tables need to be reorganized, it will be ignored by the generator.

The transactional integrity property, set to Yes by default, allows all Transactions and Procedures to be generated with transactional integrity; if otherwise desired, it can be changed to No.

If we choose the Java generator, we have for example the Compiler Options property; here we can include any valid property for the Java compiler, depending on its version.

For more information about valid properties, we recommend accessing the information in Oracle's documentation website.
https://docs.oracle.com/javase/9/tools/javac.htm#JSWOR627 .

These other properties (Reorganization Option / Create Database Option) determine whether the database creation/reorganization process will have user intervention or will take other actions in the reorganization.

The possible values are as follows:

-nogui is the default, and determines that no user intervention is allowed in these processes. For example, in case of running on a platform other than Windows that does not have a graphical interface, an error may occur during reorganization. In this way, with the -nogui command we avoid the need to use the graphical interface and solve the error.
-force. In GeneXus, we have .Gen and .Exp extension files; .Gen files are used to execute the reorganization, and .Exp files are used to control if the reorganization was executed or not. With this command, the control over these files is not performed, and the creation/reorganization of the database is forced to be performed in any case. Its use is not recommended.
-noverifydatabaseschema does not verify the database schema in the process of database creation/reorganization. It can be useful, for example, when there is a reorganization error in the schema verification process.
-recordcount: this parameter will count the records that will be created/reorganized in all tables, display the results and stop. Instead of the reorganize button, a continue button will appear. This can be used to estimate how long the task may take.
-ignoreresume. Let's suppose a reorganization was executed and failed. Then some changes are made to transactions that involve updating the database. After executing it, a new reorganization is required and another error occurs.
That is because GeneXus performs the reorganizations consecutively. If one reorganization fails and another reorganization is attempted, the previous reorganization will still be missing.
GeneXus provides routines to continue from the last execution of a reorganization before a failure.
To avoid running these validation routines and run the reorganization process from the beginning, enter the -ignoreresume command.
-donotexecute is useful for environments where you need the creation/reorganization to be generated but not executed and, once the impact analysis is done, you want to perform a complete compilation and implement the creation or reorganization and application programs somewhere else.

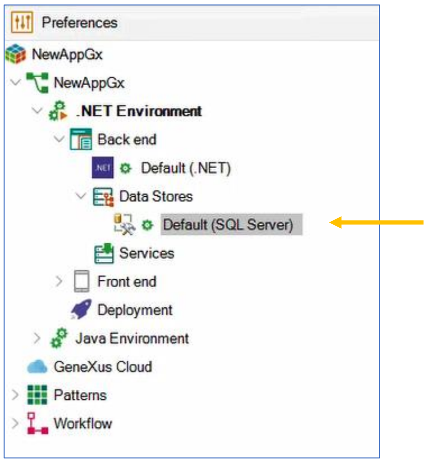
It should be noted that the commands can be combined by separating each option

with a blank space. For example, if you want to combine the nogui and force options it would look like this.
Other configurable properties are about the execution of our application. For example, we can set whether the prototyping will be in the cloud and in case of selecting yes, what the server URL will be.

We can also modify the Web server to be used, among other options.

GeneXus

Some properties of the database generator



For the database generators, the properties also depend on the chosen generator. For example, in the case of SQL Server:
We have the property Access technology to set, which allows us to configure the possible technologies for accessing the database. For example ADO.Net, used in the .NET or .NET Framework generator, or JDBC for the Java generator. The properties enabled to be configured in Connection Information will depend on the selected one.

In this section (Connection Information), we have for example the Database name property, which is used when the environments need database names. It indicates the name of the database that will contain the tables and indexes of the application.

We also have the Server Name property, which specifies the default name that will identify the database server. If this parameter is not specified, the generated programs will display a message on the screen requesting the system name to be entered at the time of establishing the connection.
We can also specify the port that the instance will use to communicate and exchange data with the applications.

The Connect to server property allows controlling when the generated programs establish a connection with the database. The default option is “At first request”, which means that the application will try to establish the connection immediately after sending the first request to the database server.

The other option is “At application startup.” In this case, the application will try to establish the connection as part of the initialization process, before the main object is displayed.
An example to understand the difference could be when we have a procedure that performs calculations without a connection to the database, and if a certain condition is met, it calls another procedure that does make a connection to the database. With “At first request” a connection is established when the application is initialized, without knowing if the second procedure is going to be used or not. And with “At application startup” a connection with the database is established only if the condition is met and the second procedure is called, which is the one that requires a connection with the server. This last option is recommended in applications that depend entirely on the database server.

The Database schema property allows pointing to a specific database schema.
A database schema is the one that describes its structure; it is a model or architecture of how our data will look like. In a relational database, the schema defines its tables, its fields in each table and the relationships between each field and each table.

These are some of the configurable properties of our KB. As we can see, there are many more whose functionality is not mentioned in this video.

If we select a property and press the “F1” key, we are taken to the GeneXus Wiki and specifically to the description and usage of the chosen property. We can also visit the wiki and search by the name of the property. Note that the “F1” key shortcut may not be available for some properties yet.

GeneXus™

training.genexus.com
wiki.genexus.com