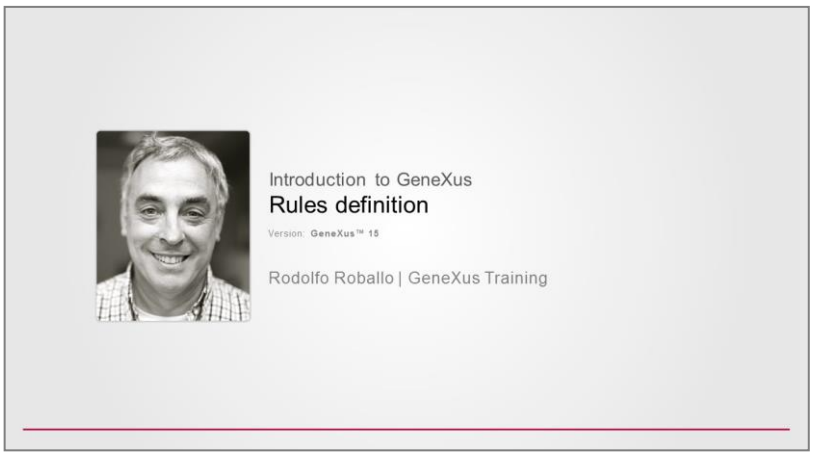
Definición de reglas



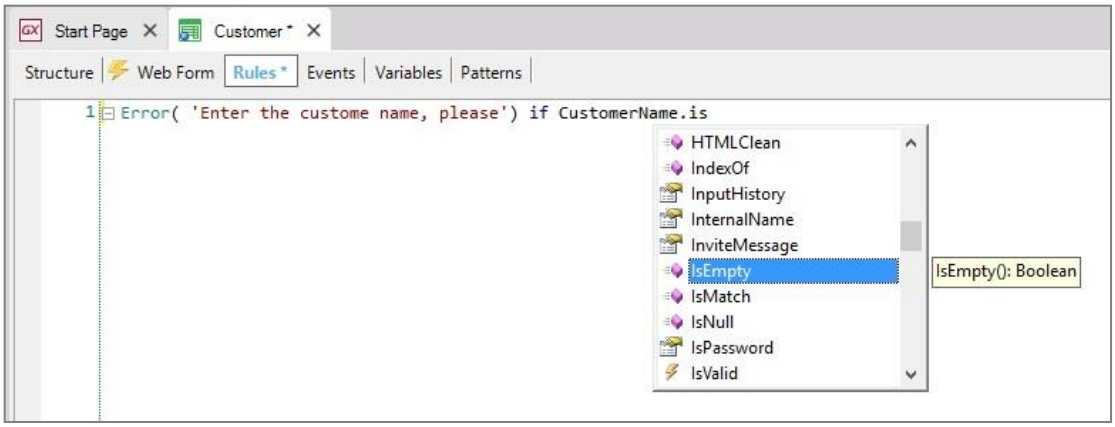
Además de todos los controles automáticos que GeneXus incluye en las aplicaciones que genera, hay ciertos controles específicos que los usuarios desean efectuar.

En las transacciones, las reglas que deben cumplirse, o los controles que nos solicitan efectuar, se definen en la sección de Rules.

Si por ejemplo un requisito es **no permitir almacenar clientes sin nombre….** contamos con una regla llamada **Error,** que nos permitirá controlar esto.

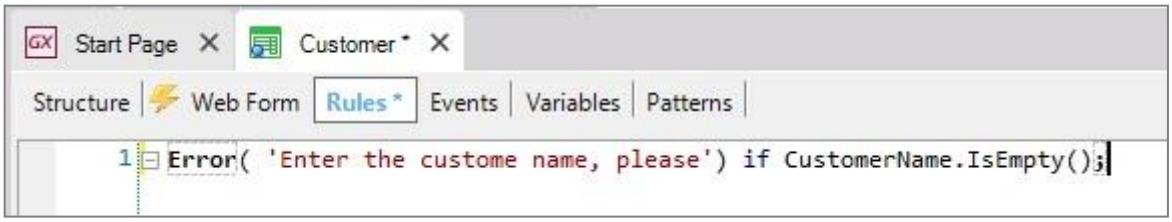
Escribimos "Error", paréntesis, y entre comillas digitamos el texto que queremos que se visualice, cuando el usuario intente dejar un nombre de cliente sin completar… cerramos el paréntesis…. y solamente nos está faltando indicar **cuál es la condición que se debe dar** para que se despliegue el texto.

La condición es que el atributo **CustomerName** esté vacío… entonces escribimos "if CustomerName", punto,



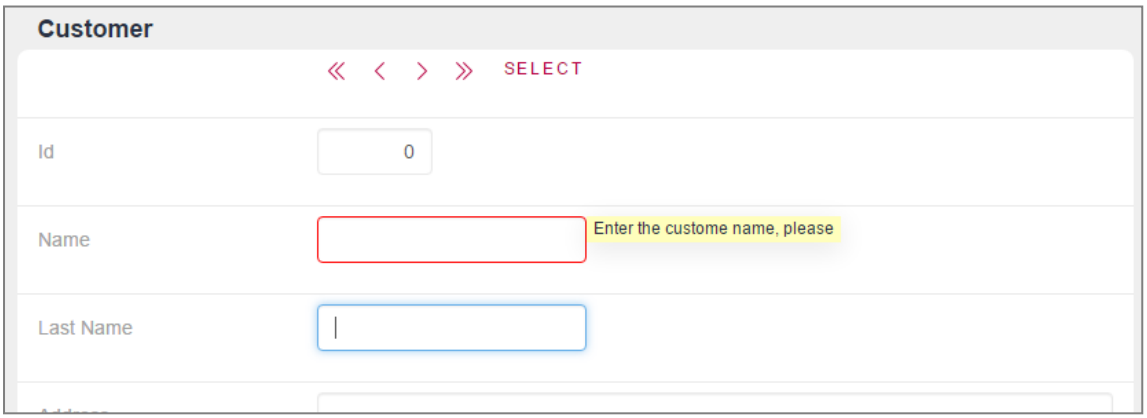
y aquí seleccionamos: IsEmpty

Toda regla que definamos debe finalizar con un punto y coma, así que lo incluimos.



Salvamos…. y presionamos F5, para ver en ejecución el funcionamiento de esta regla definida.

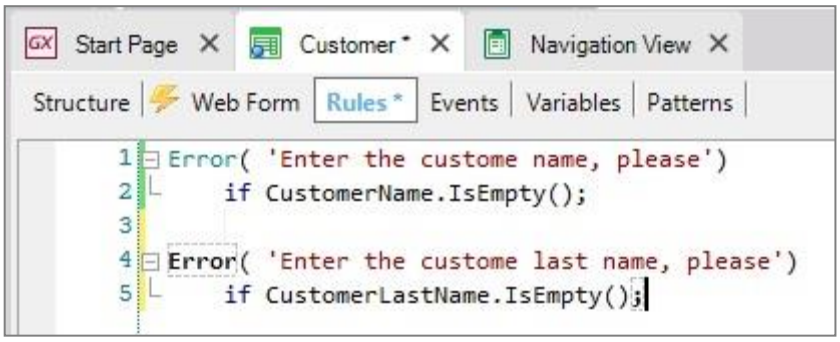
Ejecutamos la transacción Customer y si dejamos el nombre del cliente vacío y salimos del campo,



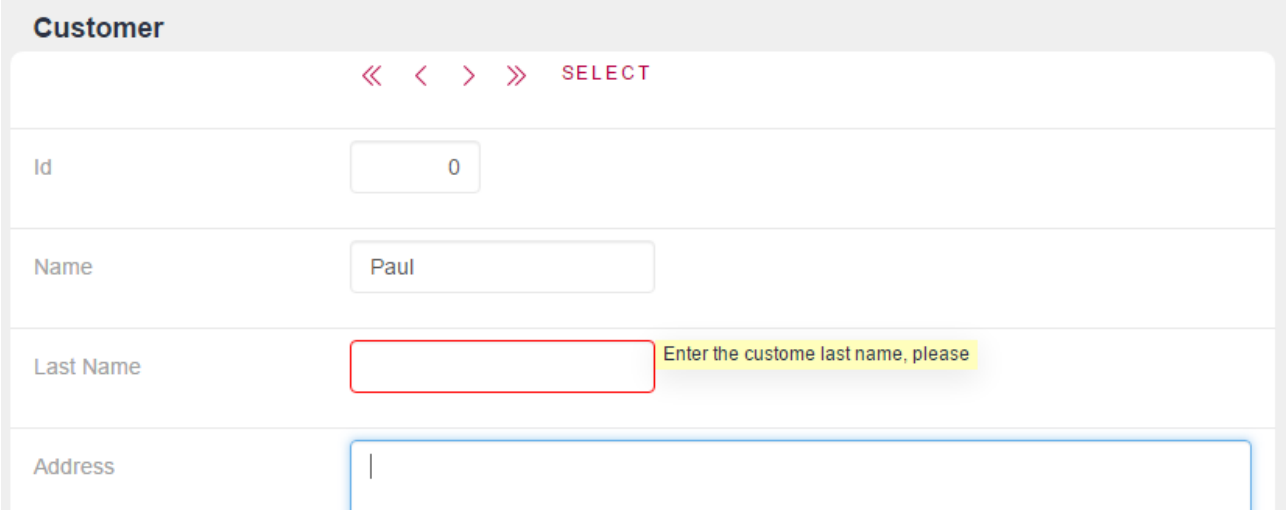
aparece el texto que definimos.

La regla **Error, aunque permite pasar al campo siguiente incluso cuando la condición se cumple, no permitirá grabar** … [probarlo] Este es el comportamiento por defecto. Puede modificarse para que la regla error nunca permita editar el siguiente campo. Es a través de una propiedad que no veremos aquí.

Si fuera requisito impedir que el apellido del cliente quede vacío, habría que definir una regla análoga. Así que copiemos y peguemos la definición de esta regla… modifiquemos el texto name por **lastname** y el atributo involucrado también:



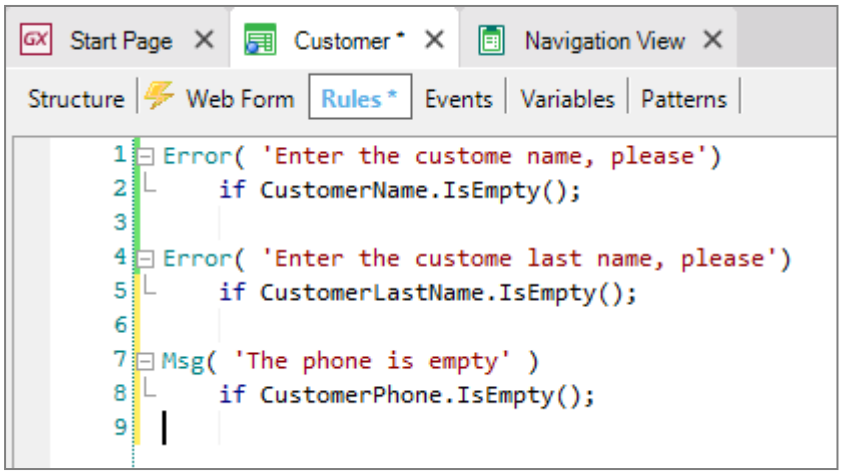
Pulsamos F5… ejecutamos Customer… dejamos el nombre del cliente vacío… sale el error asociado a que el nombre se dejó vacío…… ingresamos Paul … intentamos dejar el apellido vacío… y sale el error asociado a que el apellido se dejó sin completar.



Ahora bien… contamos con otra regla que tiene prácticamente la misma sintaxis que **Error**… su nombre es **Message…** y la única diferencia que presenta con respecto a **Error** es que de cumplirse la condición, sale el mensaje como aviso o advertencia, **y se podrá grabar el registro de todas formas**.

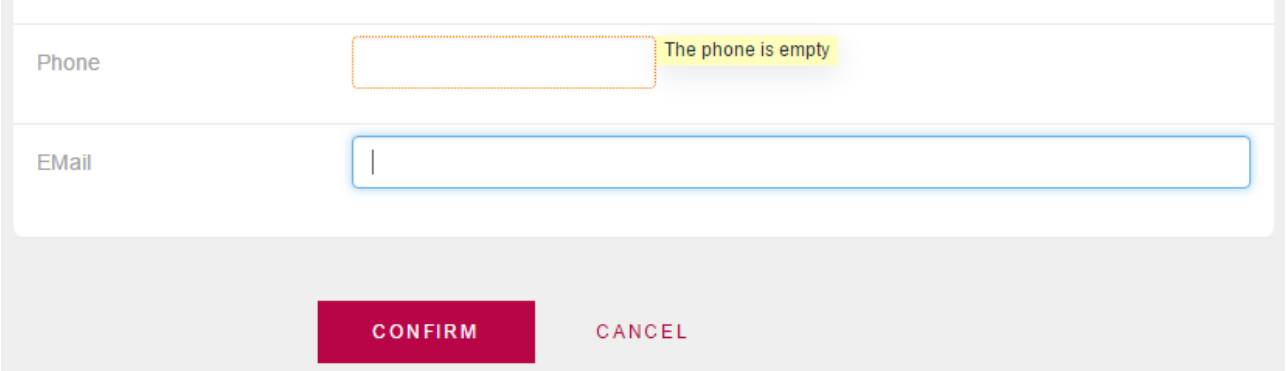
Si por ejemplo queremos avisar que han dejado sin ingresar el teléfono del cliente, pero sin obligar a ingresarlo, podemos definir una regla **Message**, paréntesis, luego el texto entre comillas simples (o dobles)… 'The phone is empty'

cerramos el paréntesis… y a continuación definimos la condición para que se ejecute la regla: "if CustomerPhone" … punto… IsEmpty. Y punto y coma para finalizar la definición de la regla.



Presionemos F5 para probar esta funcionalidad…

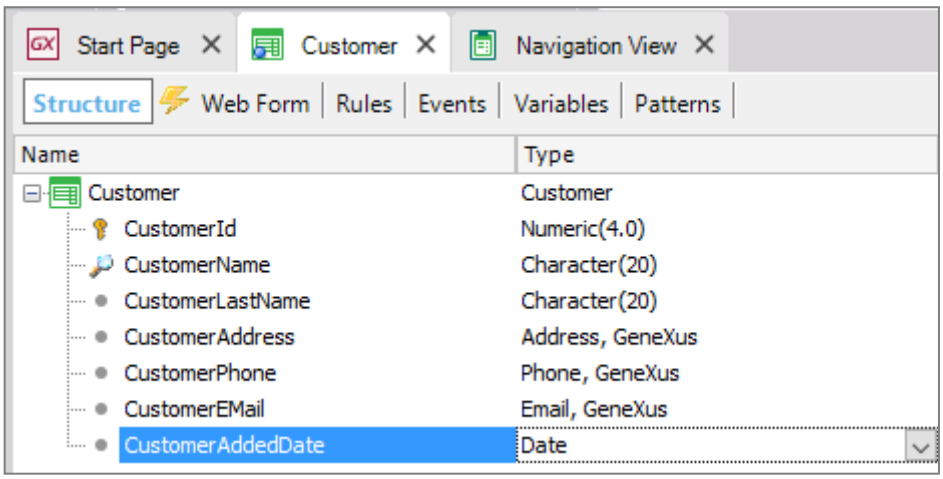
Vemos que si dejamos sin ingresar el teléfono e intentamos salir del campo,



sale el mensaje que definimos, y si ahora queremos grabar, lo conseguimos.

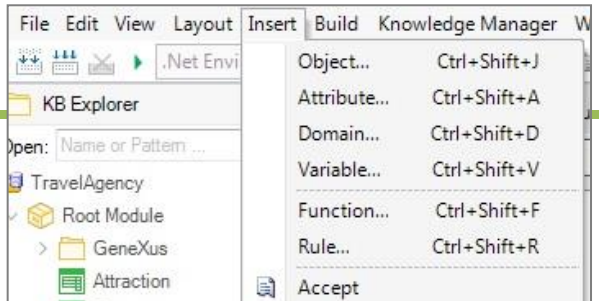
Ahora bien….supongamos que los usuarios de la agencia de viajes nos piden **que les interesa almacenar para cada cliente la fecha en la cual es dado de alta**. Llamamos "dar de alta" a la operación de insertar.

Necesitamos entonces crear un nuevo atributo en la transacción Customer, para almacenar dicha fecha. Definimos **CustomerAddedDate** … de tipo Date…

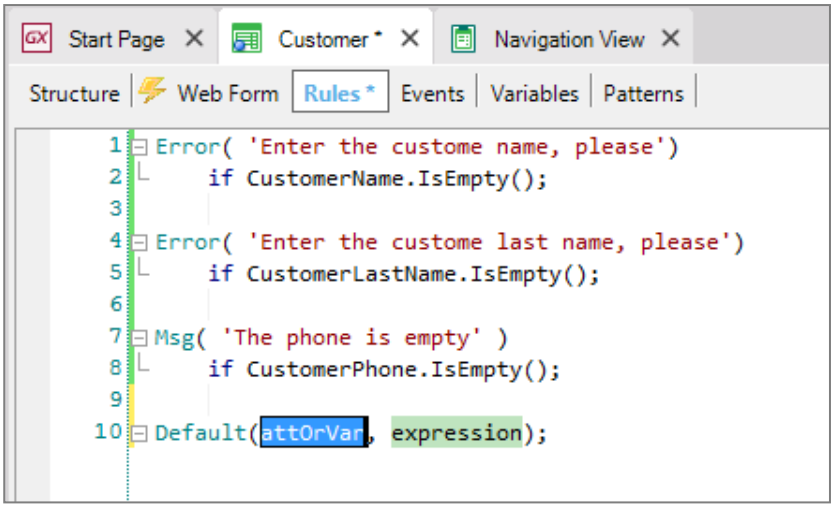


y nos restaría asignarle automáticamente la fecha del día.

Vayamos a la sección de Rules…. y contamos con una regla llamada **Default**

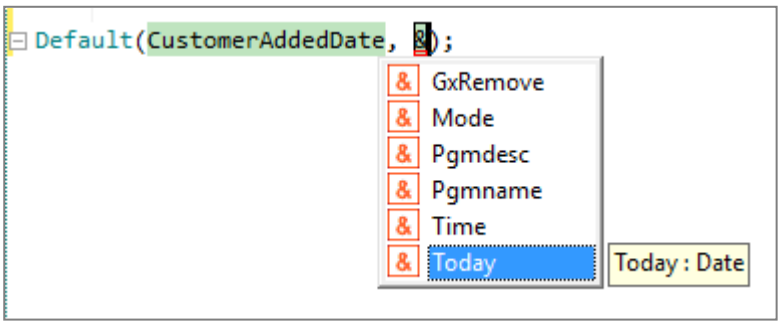


Esta regla nos permite inicializar a un atributo o variable con un valor:



De esta manera se nos insertó la sintaxis de la regla Default, y ahora vamos a sustituir dentro de los paréntesis, al atributo que queremos inicializar, que es **CustomerAddedDate** y el valor con el cual lo queremos inicializar, que es la fecha de hoy.

“Ampersand

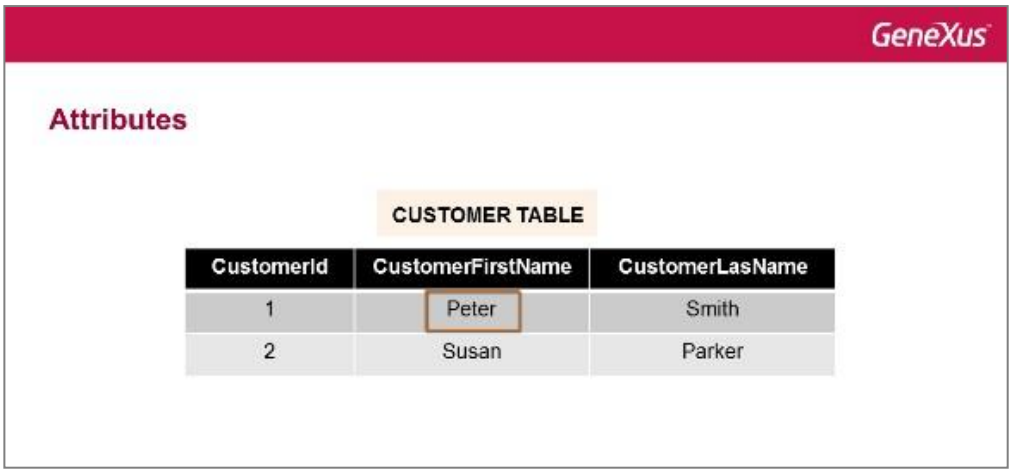


today” es una variable predefinida, que siempre tiene cargada la fecha del día como para utilizarla.

```
Default(CustomerAddedDate, &Today);
```

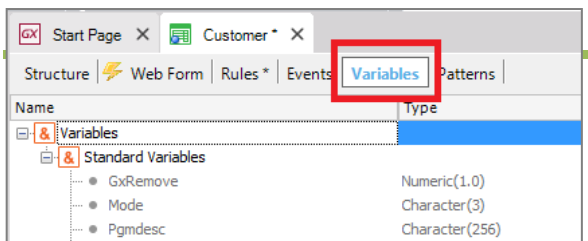
Una variable consiste en un espacio de memoria, posee un nombre simbólico, y un tipo de datos que puede almacenar (texto, números, fechas, etc.) Esa variable tiene un determinado valor almacenado (guardado). El nombre de la variable es la forma usual de referirse a dicho valor almacenado.

Un **Atributo** en cambio**, es un** Valor almacenado a nivel físico en la base de datos

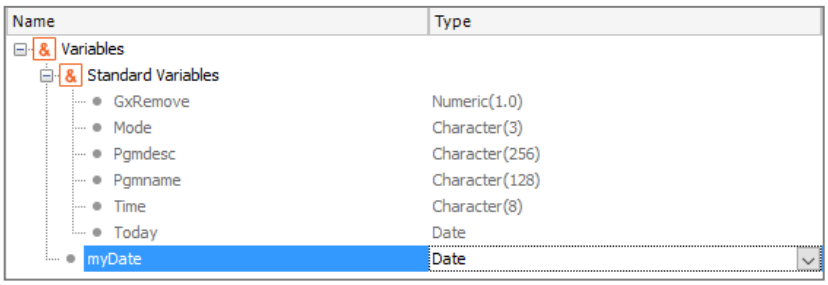


La mayoría de los objetos GeneXus permiten definir variables. Esas variables son **locales**, lo cual significa que solamente pueden ser utilizadas dentro de dicho objeto. Para referenciar a una variable debemos utilizar el símbolo "&". Por ejemplo &Total.

Si vamos por ejemplo al selector *Variables* dentro de una transacción veremos que ya hay un conjunto de variables definidas. Son variables del sistema: como por ejemplo, &Today, &Mode, etc. En particular la variable &Today permite obtener la fecha actual tomada del sistema.



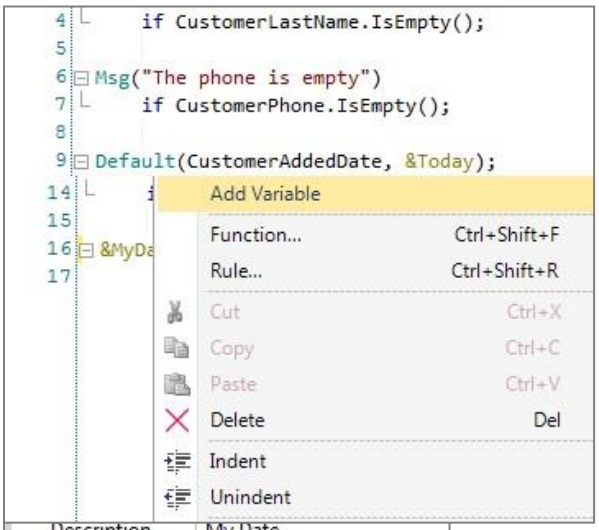
Más allá de estas variables del sistema, el desarrollador también puede definir sus propias variables (variables de usuario). Por ejemplo, una variable myDate de tipo Date:

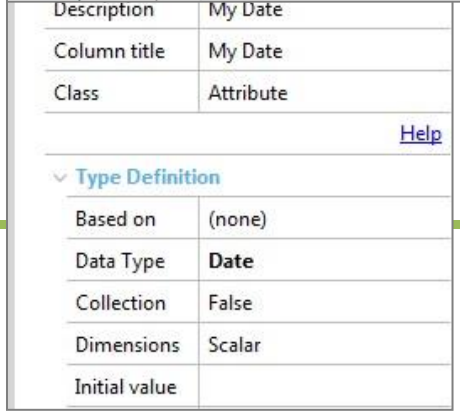


Al momento de definir variables, se puede optar por:

1) Definirlas a través del selector Variables presente en todos los objetos GeneXus, como acabamos de hacerlo.

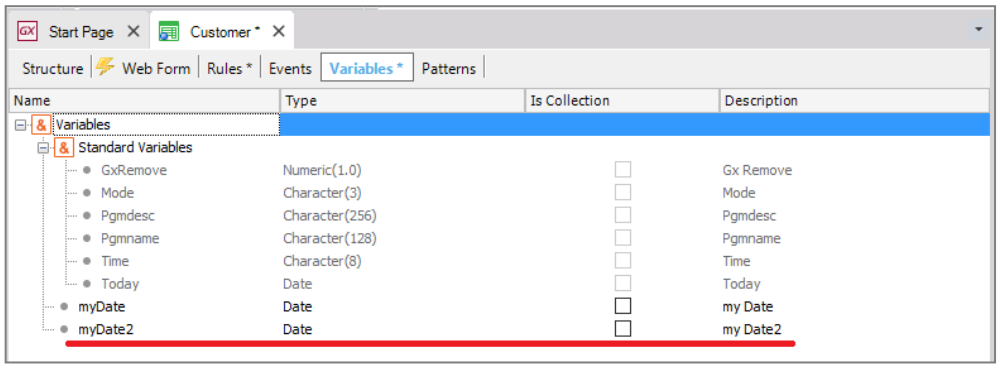
2) Definirlas al momento de usarlas desde el mismo lugar donde se las necesita. Por ejemplo: escribir el símbolo ampersand que identifica que lo que sigue es el nombre de una variable y luego su nombre, y por último posicionar el mouse sobre el nombre que se le dio a la variable y presionar el botón derecho del mouse para elegir del menú contextual: **Add Variable**.



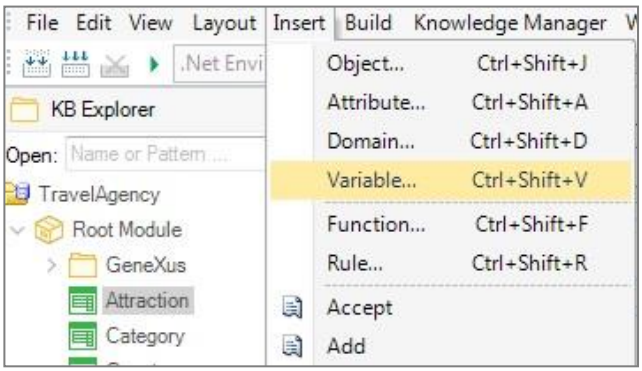


Vemos que ya nos edita las propiedades de la variable. Allí podemos asignarle su tipo de datos.

Si vamos al selector de Variables , allí la vemos.



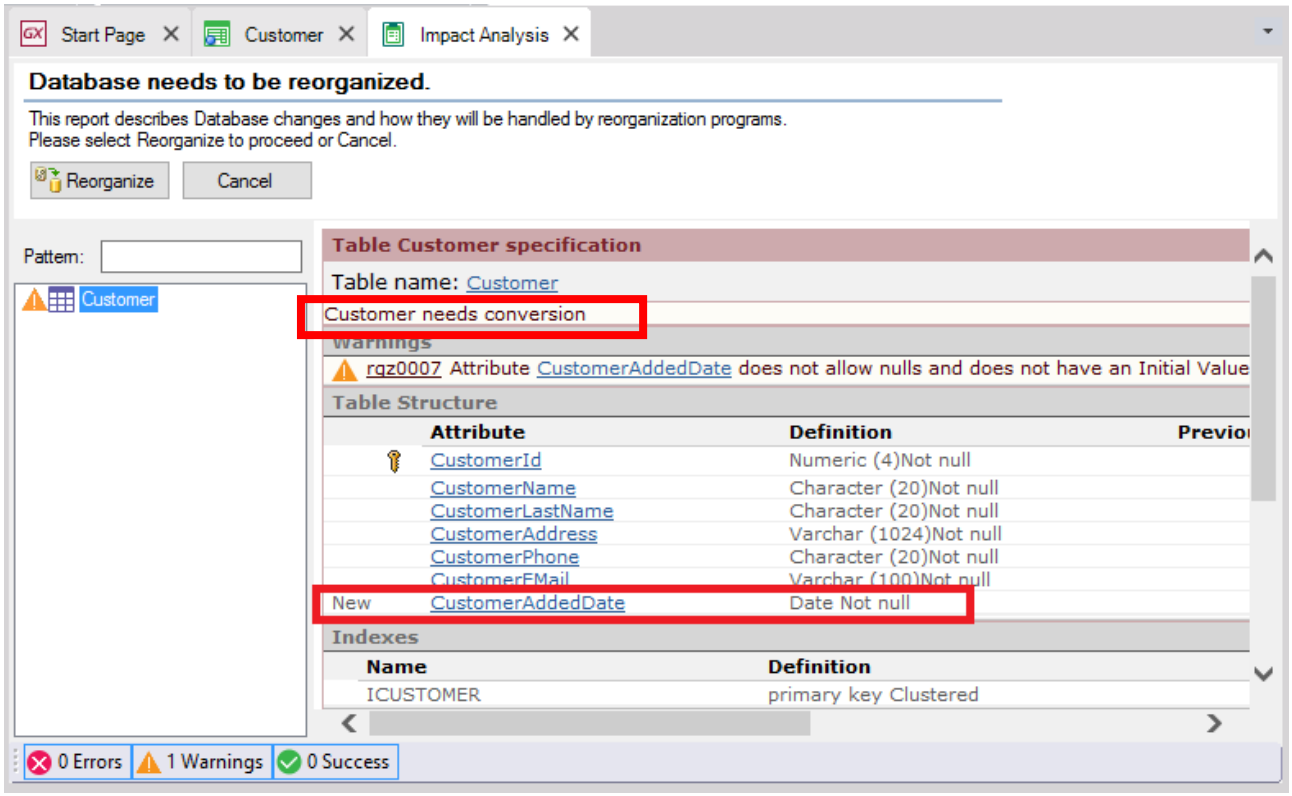
3) La tercera opción es seleccionar la opción **Insert** de la menubar y luego **Variable… y New Variable…**



En el ejemplo que se muestra, a través de la regla Default, estamos asignando por defecto la fecha actual al atributo CustomerAddedDate

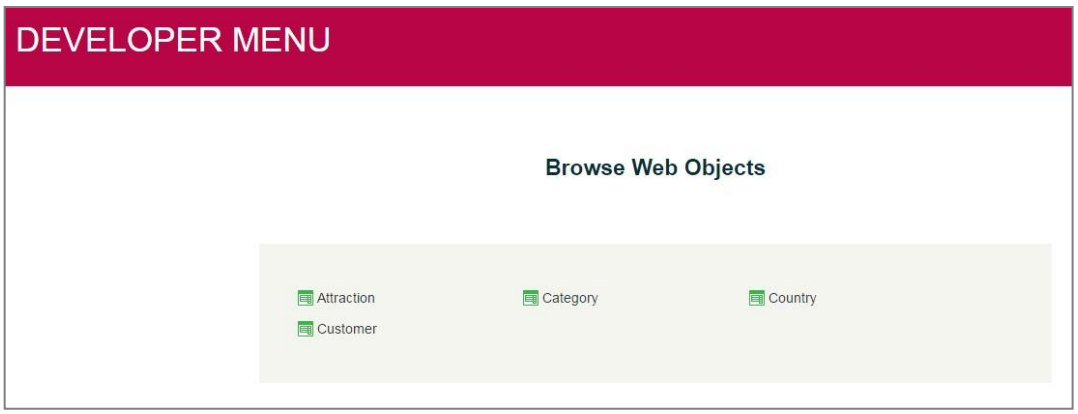
Ahora vamos a grabar… y presionamos F5.

Se nos avisa que se va a agregar el nuevo atributo **CustomerAddedDate** a la tabla CUSTOMER:

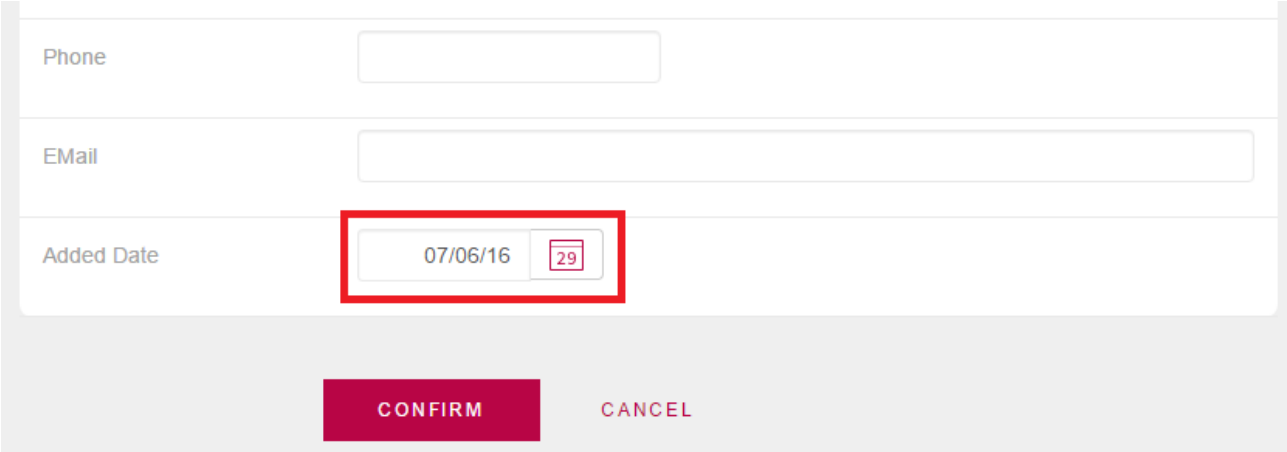


Procedemos a reorganizar…

y nuevamente contamos con la aplicación para ejecutarla.



Entramos a Customer…

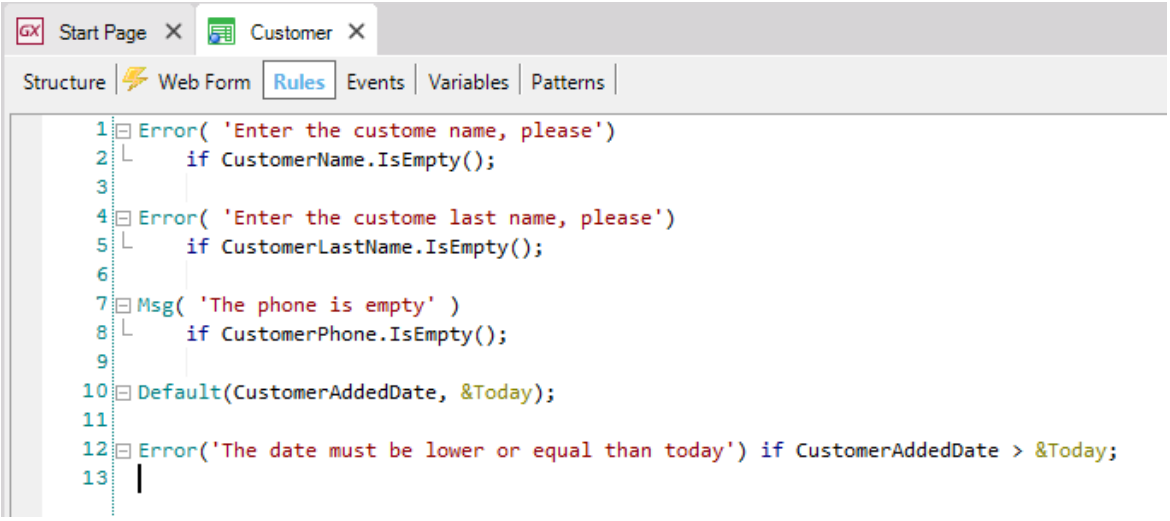


y ya podemos percibir el nuevo atributo “fecha de inserción” inicializado con la fecha de hoy.

Si no hubiéramos definido la regla Default, el campo de la fecha aparecería vacío como los demás campos.

Ingresemos un cliente,… , Robert… Hill… que vive en la calle 81…. su teléfono es el 760 5100 y su mail es rhill@example.com.. Y observemos que la fecha de hoy se nos sugiere, **pero la podemos modificar**.

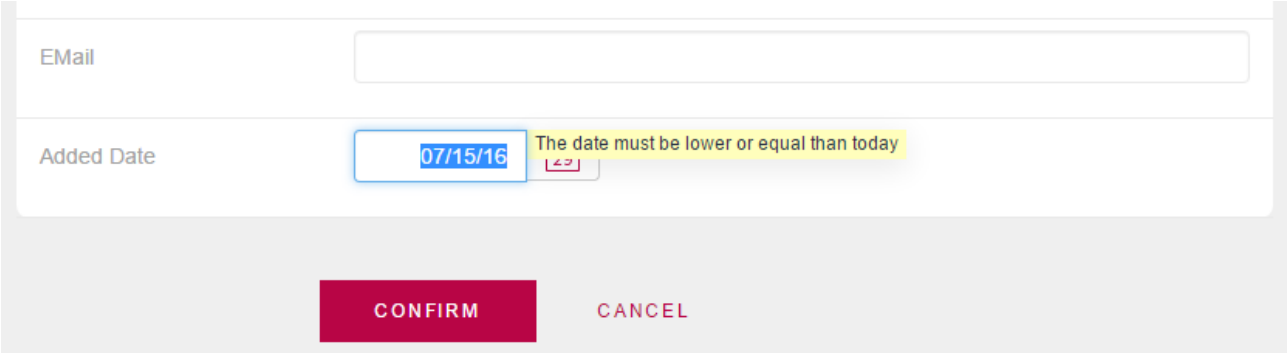
Si a los usuarios de la agencia de viajes les interesara dejar la fecha editable, **pero que controlemos que no puedan ingresar fechas futuras**… podríamos definir una regla **Error**.
Abrimos paréntesis, digitamos ‘The date must be lower or equal than today’, cerramos paréntesis … y agregamos la condición **if CustomerAddedDate > &today;**



Vamos a probar esto en ejecución, pulsamos F5…

Ingresamos a Alex… Johnson…

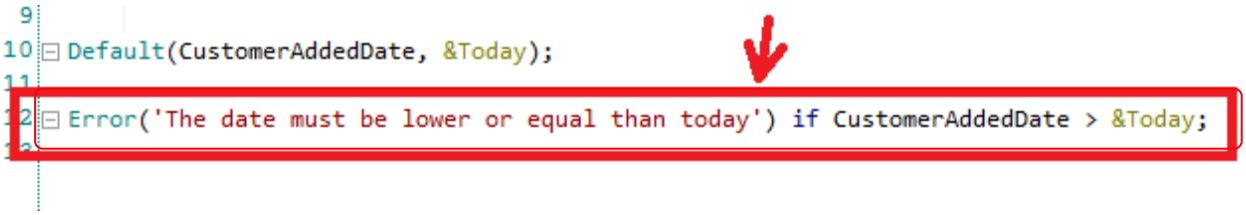
Y si intentamos poner una fecha mayor a la del día de hoy….



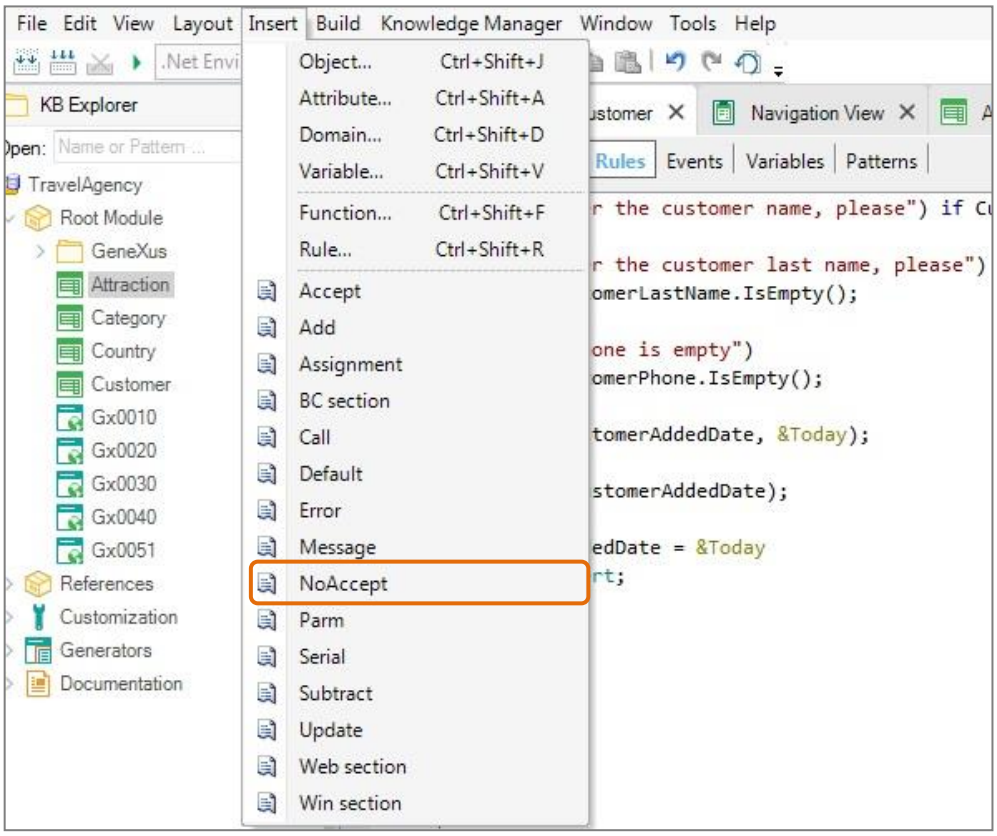
se da la condición que definimos, y sale el error asociado.

Ahora supongamos que en la agencia de viajes nos indican que la fecha de alta del cliente **no puede ser editada**, sino que debe verse deshabilitada en el formulario y grabarse tal cual la sugirió la aplicación….

Para lograr este pedido, eliminaríamos esta regla, porque ya no tiene sentido.



Y tendríamos que definir una regla… Noaccept



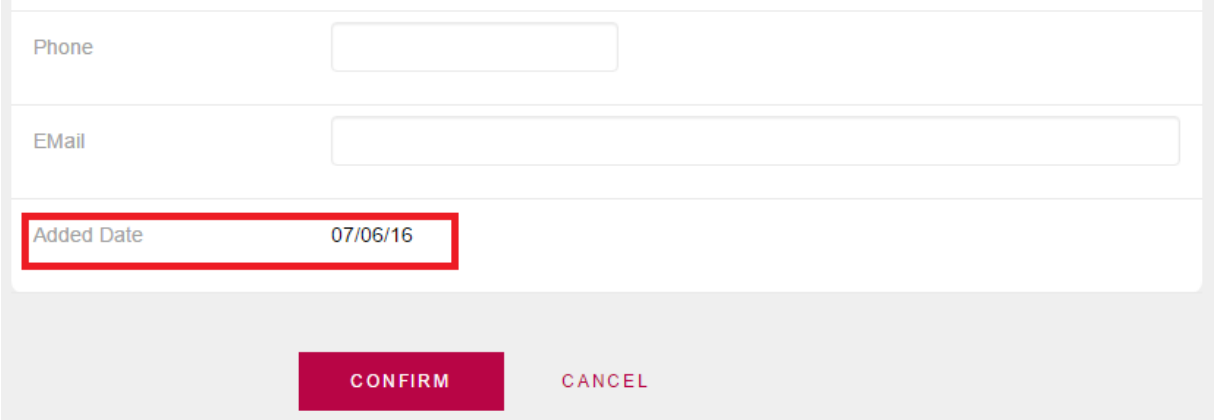
```
12 NoAccept(attOrVar) If condition;
13
```

Sustituimos dentro del paréntesis el texto que dice “attribute or variable”, por el atributo **CustomerAddedDate** y borramos “if condition”, porque queremos que esta regla se ejecute siempre.

```
12 NoAccept(CustomerAddedDate);
```

Probemos el comportamiento ahora…

Y vemos que la fecha aparece **inicializada** por la regla default **y deshabilitada** por la regla noaccept.



Muy bien...

Vimos que para inicializar el atributo CustomerAddedDate con la fecha del día, definimos esta regla Default:

```
9
10 Default(CustomerAddedDate, &Today);
11
```

Es importante saber que toda regla Default que definamos, se ejecutará **solamente cuando estemos insertando registros.**

O sea que si se consulta un cliente que ya estaba almacenado, la regla Default no se ejecutará… ya que dicho cliente **ya tiene su fecha de inserción…** y la regla Default no la sobreescribe.

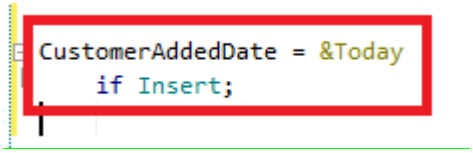
Ahora supongamos que en lugar de haber definido la regla Default hubiéramos definido esta asignación:

```
CustomerAddedDate = &Today;
```

Mediante la definición de esta regla, el atributo CustomerAddedDate **sería asignado siempre** con la fecha del día. **Esta es una regla de asignación**, y se ejecutaría siempre, independientemente de si el usuario está insertando, actualizando, etc.

A una regla de asignación la podemos condicionar para que se ejecute solamente cuando el usuario está efectuando determinada operación específica en la base de datos, es decir una inserción, actualización o eliminación.

Vamos a hacerlo. Agregamos **if insert** :

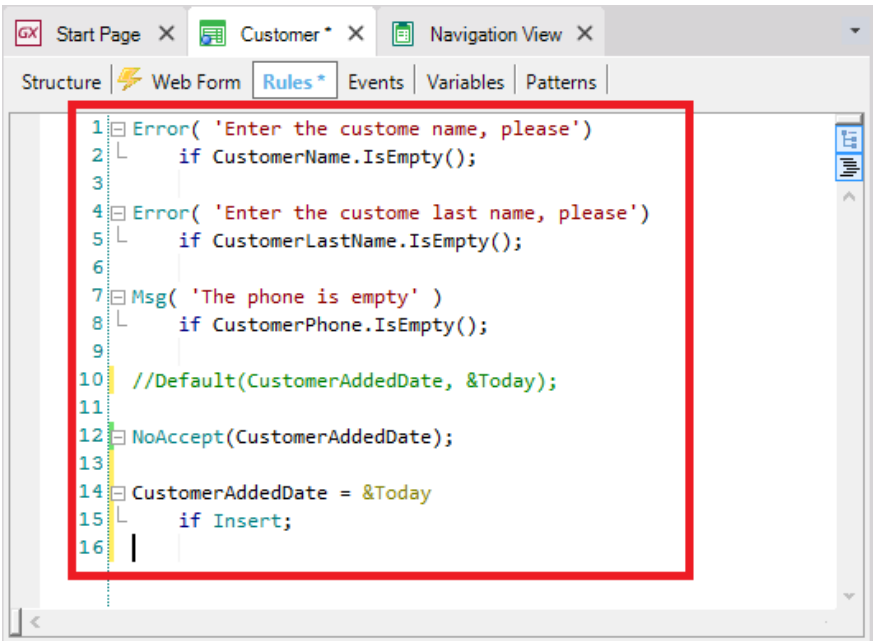


El comportamiento de esta regla definida así, será equivalente a lo que realiza la regla Default, ya que ahora hemos condicionado que la asignación se realice solamente si se está insertando un registro, y es lo que hace la regla Default.

Así como se puede condicionar una regla con **if insert**, contamos con la posibilidad de condicionar reglas a que se ejecuten **if update** o **if delete** también.

Algo que es importante observar y saber, es que **el orden en el que definimos las reglas no corresponde necesariamente al orden en el que serán ejecutadas**.

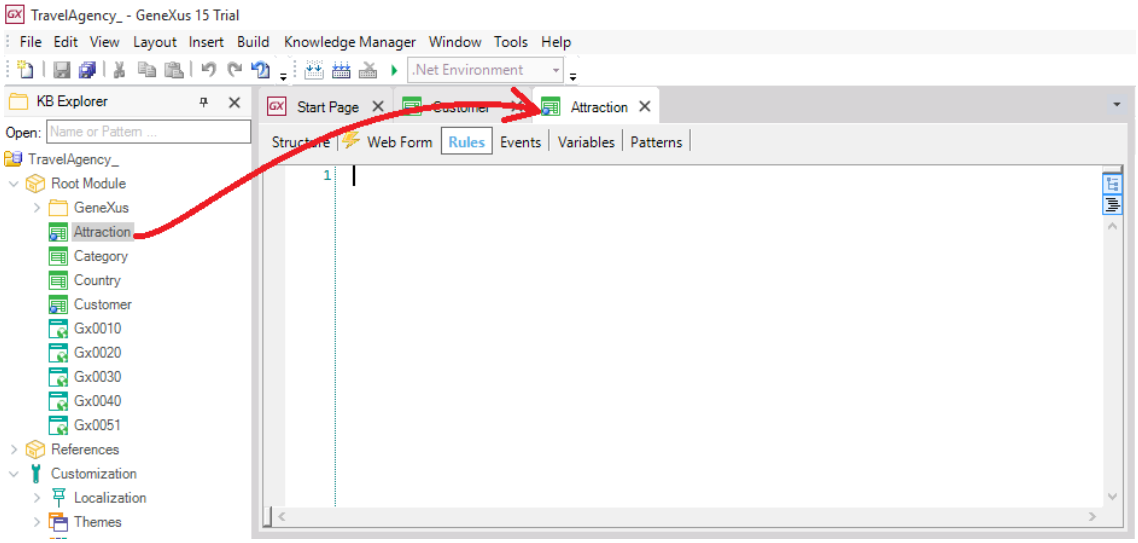
Este conjunto de reglas:



podría estar definido en cualquier otro orden y el resultado en ejecución sería exactamente el mismo, ya que GeneXus decide en qué momento debe dispararse cada una de las reglas definidas.

Para finalizar, recordemos que a cada transacción habrá que definirle de ser necesario, sus propias reglas de comportamiento.

En este caso, hemos definido reglas en la transacción de clientes, para controlar las particularidades que nos solicitaron cuidar cuando los usuarios interactúen con los datos de los clientes. Muy probablemente la agencia pretenda controlar ciertas reglas o comportamiento para las atracciones también…



o para otra transacción. Y para ello, **cada transacción cuenta con su propia sección de reglas.**

Para terminar, vamos a grabar los cambios en GeneXus server.