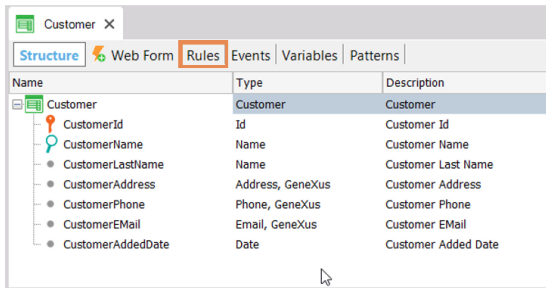


Reglas: Repaso y Client-Side Validation

GeneXus™

Rules



Name	Type	Description
Customer	Customer	Customer
CustomerId	Id	Customer Id
CustomerName	Name	Customer Name
CustomerLastName	Name	Customer Last Name
CustomerAddress	Address, GeneXus	Customer Address
CustomerPhone	Phone, GeneXus	Customer Phone
CustomerEMail	Email, GeneXus	Customer EMail
CustomerAddedDate	Date	Customer Added Date

En el curso anterior aprendimos que el objeto Transacción cuenta con una sección llamada **Rules**, en la cual son definidos los controles que se deben efectuar o las reglas que se deben cumplir para una cierta realidad.

Rules

```
Customer X
Structure | Web Form | Rules | Events | Variables | Patterns
1 Error('Enter the customer name')
2 if CustomerName.IsEmpty();
```

Customer

Navigation: << < > >> SELECT

Id:

Name: Enter the customer name ←

Last Name:

Address:

E-Mail:

Phone:

Added Date:

Buttons: CONFIRM CANCEL

Nos centramos en la regla **Error**, que impide que se almacene en la base de datos un registro mientras cierta condición se cumpla: por ejemplo, si estamos ingresando un cliente y dejamos su nombre vacío, ese cliente no se podrá insertar hasta que el usuario haya escrito su nombre.

Rules

```
Customer X
Structure | Web Form | Rules | Events | Variables | Patterns
1 | Error('Enter the customer name')
2 |   if CustomerName.IsEmpty();
3 |
4 | Msg('The phone is empty')
5 |   if CustomerPhone.IsEmpty();
```

Customer

• Data has been successfully added.

Navigation: << < > >> SELECT

Id:

Name:

Last Name:

Address:

E-Mail:

Phone: The phone is empty ←

Added Date:

CONFIRM CANCEL

Vimos también la regla **Message**, la cual únicamente informa al usuario mediante un mensaje pero permite efectuar la grabación;

Rules

```
Customer X
Structure | Web Form | Rules | Events | Variables | Patterns
1 Error('Enter the customer name')
2   if CustomerName.IsEmpty();
3
4 Msg('The phone is empty')
5   if CustomerPhone.IsEmpty();
6
7 Default(CustomerAddedDate, &Today);
```

Customer

« < > » SELECT

Id

Name

Last Name

Address

EMail

Phone

Added Date ←

CONFIRM CANCEL

la regla **Default**, que nos permite inicializar un atributo o variable con un valor cuando accedemos a la transacción en modo de inserción;

Rules

```
Customer X
Structure | Web Form | Rules | Events | Variables | Patterns
1 Error('Enter the customer name')
2   if CustomerName.IsEmpty();
3
4 Msg('The phone is empty')
5   if CustomerPhone.IsEmpty();
6
7 Default(CustomerAddedDate, &Today);
8
9 Noaccept(CustomerAddedDate);
```

Customer

« < > » SELECT

Id


Name

Last Name

Address

EMail

Phone

Added Date 08/27/20 

CONFIRM CANCEL

la regla **NoAccept**, que impide que el usuario pueda modificar un campo en el formulario: lo muestra deshabilitado,

Rules

Country * X

Structure | Web Form | **Rules *** | Events | Variables | Patterns

```
1 Serial(CityId, CountryLastLine, 1);
2
3
4
5
```

Country * X

Structure | Web Form

Name

- Country
 - CountryId
 - CountryName
 - CountryLastLine
- City
 - CityId
 - CityName

Country

« < > » SELECT

Id

Name

Last Line 2

City

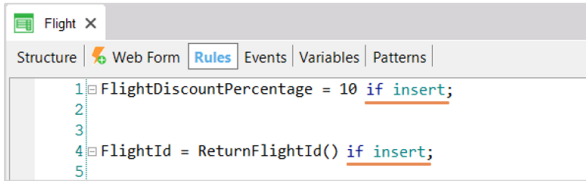
Id Name	
X 1	São Paulo
X 2	Rio de Janeiro
<input type="text" value=""/>	<input type="text" value=""/>
0	
0	

[New row]

CONFIRM CANCEL

y por último la regla Serial, que sirve para autonumerar un segundo nivel, o tercero, u otro nivel anidado de una transacción.

Rules



```
1 FlightDiscountPercentage = 10 if insert;  
2  
3  
4 FlightId = ReturnFlightId() if insert;  
5
```

if insert

if update

if delete

Estudiamos también que mediante las reglas podemos definir asignaciones de valores o invocar objetos, y que también podemos condicionarlas para que se ejecuten solo cuando se está insertando, modificando o eliminando.

Rules

```

1 FlightDiscountPercentage = 10 if insert;
2
3
4 FlightId = ReturnFlightId() on BeforeInsert;
5

```



Además, si el momento elegido por GeneXus para ejecutar una regla no es el que necesitamos, podemos indicarle en qué momento preciso queremos que se ejecute, utilizando los eventos de disparo.

Execution of rules



Recordemos que todas estas reglas que hemos estudiado, son validadas tanto en el cliente web como en el servidor.

La validación que ocurre en el cliente es llamada **Client-Side Validation**, y su objetivo es brindar una buena experiencia al usuario, haciéndole sentir que la aplicación está todo el tiempo interactuando con él; pero es el servidor quien realmente valida que toda la información enviada sea consistente y no viole la seguridad del sistema, y es el único que puede operar sobre la base de datos.

The screenshot displays the GeneXus IDE interface. On the left, the KB Explorer shows a project named 'TravelAgency'. The main Start Page features 'TOP NEWS' with two articles: 'DigitalSignerApplet' and 'GXWS-Launcher'. At the bottom, there are two prompts: 'Are you ready to create your own Knowledge Base? OK, go ahead!' and 'Create your own Knowledge Base by using some of our samples and make awesome apps!'. On the right, the Properties window is open, showing the 'User interface' section. A table lists various UI properties and their default values. The 'Client side validation behaviour' section is highlighted, showing options for 'Stop on error', 'Validation message position', 'Validation message overlap adjacent controls', and 'Validation message display'. The 'Defaults' section shows 'Report output' as 'Only To File' and 'Exposed namespace' as 'TravelAgency'.

User interface	
Web interface	
Default Theme	Carmine
Web User Experience	Smooth
Default Web Form Editor	Abstract Layout
Web Form Defaults	Responsive Web Design
Default Master Page	RwdMasterPage
Datepicker image	calendar
Delete grid row UI	Alternating images
Delete row image	DeleteRow
Undelete row image	UndeleteRow
Prompt image	(none)
Delete column position in grids	Leftmost column
Delete column position in free-style grids	Top left
Delete column tooltip text	
Delete column title	
Client side validation behaviour	
Stop on error	No
Validation message position	Right
Validation message overlap adjacent controls	Yes
Validation message display	One at a time
Defaults	
Report output	Only To File
Exposed namespace	TravelAgency

Hasta aquí hemos visto el comportamiento por defecto de estas reglas. Pero es interesante saber que contamos con un grupo de propiedades llamado **Client Side Validation Behaviour** que encontramos a nivel de la versión, el cual nos permite personalizar el comportamiento y los mensajes de las reglas de muchas formas, permitiendo que la aplicación sea más atractiva para usuarios finales.

Stop on error property

Customer

« < > » SELECT

Id: 0

Name: Tom

Last Name: |

Address:

Phone:

Email:

Added Date: 08/30/20

CONFIRM CANCEL

Client side validation behaviour	
Stop on error	Yes
Validation message position	Right
Validation message overlap adjacent controls	Yes
Validation message display	One at a time

Cuando utilizamos la regla Error, vemos que aunque la transacción no permite grabar, sí permite pasar al siguiente campo, luego de haber mostrado el mensaje de error. Si nos interesara que en caso de dispararse un error, se mantenga el foco en el control y se obligue al usuario a corregir el valor para poder pasar al siguiente campo, debemos cambiar el valor de la propiedad **Stop on error** para Yes.

Validation message position property

Customer

« < > » SELECT

id

Name Enter the customer name

Last Name

Address

Phone

Email

Added Date 08/30/20

CONFIRM CANCEL

Client side validation behaviour	
Stop on error	No
Validation message position	Bottom
Validation message overlap adjacent controls	Yes
Validation message display	One at a time

También podremos definir el lugar donde va el mensaje con respecto al campo, utilizando la propiedad **Validation Message Position**, que cuenta con los valores *Right* (que es el valor por defecto), *Left*, *Top* y *Bottom*.

Validation message overlap adjacent property

Customer

« < > » SELECT

Id

Name Enter the customer name

Last Name

Address

Phone

Email

Added Date 08/30/20

CONFIRM CANCEL

Client side validation behaviour	
Stop on error	No
Validation message position	Bottom
Validation message overlap adjacent controls	Yes
Validation message display	Yes
	No

En caso de que el mensaje se muestre solapado a algún otro control, también debe configurarse la propiedad **Validation message overlap adjacent controls**, que controla si los mensajes deben superponerse o no.

Validation message display property

Customer

« < > » SELECT

Id

Name

Last Name

Address

Phone The phone is empty

Email

Added Date 08/30/20

CONFIRM CANCEL

Client side validation behaviour

Stop on error	No
Validation message position	Right
Validation message overlap adjacent controls	Yes
Validation message display	One at a time

La última propiedad del grupo es **Validation message display**, que cuenta con los valores *One at a time* (que es el valor por defecto) y cuyo objetivo es mostrar siempre el último mensaje de validación que activa la aplicación,

Validation message display property

The screenshot shows a form titled "Customer" with the following fields and validation messages:

- Id:** 0
- Name:** Enter the customer name
- Last Name:** (empty)
- Address:** (empty text area)
- Phone:** The phone is empty
- Email:** (empty)
- Added Date:** 08/30/20

At the bottom, there are two buttons: **CONFIRM** and **CANCEL**.

A configuration panel titled "Client side validation behaviour" is open, showing the following settings:

Client side validation behaviour	
Stop on error	No
Validation message position	Right
Validation message overlap adjacent controls	Yes
Validation message display	All at once

y *All at once*, que muestra al mismo tiempo cada mensaje de validación que debería estar en la pantalla.

The screenshot shows the GeneXus IDE interface. The 'Build' menu is open, and the 'Rebuild All' option is highlighted with a red arrow. The main workspace displays a list of applets: 'Applet', 'GXWS-Launcher', and 'PrintingApplet (printingapplet.jar)'. The 'Applet' entry is expanded, showing its description: 'Digital (electronic) signing applet of GeneXus Flow. Deprecated since GeneXus 16 Upgrade 4.' The 'GXWS-Launcher' and 'PrintingApplet' entries also show their descriptions and deprecation status. At the bottom of the IDE, there are two promotional messages: 'Are you ready to create your own Knowledge Base? OK, go ahead!' and 'Create your own Knowledge Base by using some of our samples and make awesome apps! Create from Sample >'

Como estas propiedades modifican el comportamiento relativo a la interacción en los forms, se deben volver a generar todos los objetos que tienen form para que los cambios sean efectuados. Para ello, debe utilizar la opción **Rebuild All**, que genera absolutamente todos los objetos.

En el siguiente video agregaremos algunas reglas interesantes a las ya conocidas y en otros estudiaremos cómo se evalúan para determinar el orden de ejecución, así como profundizaremos un poco más en los eventos de disparo.

GeneXus™

training.genexus.com
wiki.genexus.com
training.genexus.com/certifications