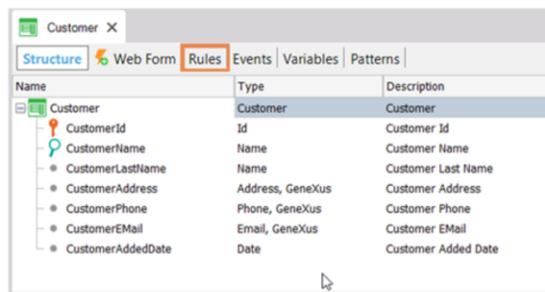


# Regras: Revisão e Client-Side Validation

*GeneXus™*

# Rules



No curso anterior aprendemos que o objeto Transação conta com uma seção chamada **Rules**, na qual são definidos os controles que devem ser efetuados ou as regras que devem ser cumpridas para uma determinada realidade.

# Rules

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

Customer

Navigation: << < > >> SELECT

id: 0

Name:  Enter the customer name ←

Last Name:

Address:

E-mail:

Phone:

Added Date: // 25

CONFIRM CANCEL

Nos concentramos na regra **Error**, que impede que um registro seja armazenado na base de dados enquanto uma determinada condição é cumprida: por exemplo, se estamos inserindo um cliente e deixamos seu nome vazio, esse cliente não poderá ser inserido até que o usuário tenha escrito seu nome.

# 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.

« < > » SELECT

id

Name

Last Name

Address

E-mail

Phone  The phone is empty ←

Added Date

CONFIRM CANCEL

Vimos também a regra **Message**, a qual somente informa ao usuário mediante uma mensagem mas permite efetuar a gravação;

# Rules

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

Customer

« < > » SELECT

Id

Name

Last Name

Address

Email

Phone

Added Date   ←

CONFIRM CANCEL

a regra **Default**, que nos permite inicializar um atributo ou variável com um valor quando acessamos à transação em modo de inserção:

## Rules

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

a regra **NoAccept**, que impede que o usuário possa modificar um campo no formulário: mostra-o desativado.

# 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

City

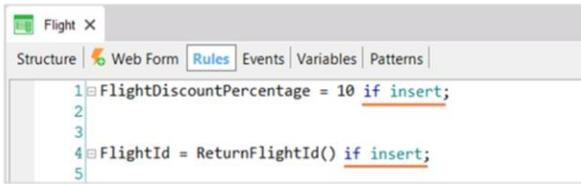
Id Name	
X 1	São Paulo
X 2	Rio de Janeiro
<input type="text" value=""/>	<input type="text" value=""/>
0	
0	

[New row]

CONFIRM CANCEL

e por último a regra **Serial**, que serve para autonumerar um segundo nível, ou terceiro, ou outro nível aninhado de uma transação.

# Rules



```
Flight X
Structure | Web Form | Rules | Events | Variables | Patterns
1 FlightDiscountPercentage = 10 if insert;
2
3
4 FlightId = ReturnFlightId() if insert;
5
```

*if insert*

*if update*

*if delete*

Estudamos também que através das regras podemos definir atribuições de valores ou invocar objetos, e que também podemos condicioná-las para que se executem somente quando se está inserindo, modificando ou eliminando.

# Rules

```

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

```



Além disso, se o momento escolhido por GeneXus para executar uma regra não é o que necessitamos, podemos indicar-lhe em que momento exato queremos que se execute, utilizando os eventos de disparo.

## Execution of rules



Recordemos que todas estas regras que temos estudado, são validadas tanto no cliente web como no servidor.

A validação que ocorre no cliente é chamada **Client-Side Validation**, e seu objetivo é proporcionar uma boa experiência ao usuário, fazendo-o sentir que a aplicação está sempre interagindo com ele; mas é o servidor quem realmente valida que toda a informação enviada seja consistente e não viole a segurança do sistema, e é o único que pode operar sobre a base de dados.

The screenshot shows the GeneXus IDE interface. On the left, the 'KB Explorer' shows the project structure for 'TravelAgency'. The main 'Start Page' features a 'TOP NEWS' section with two articles: 'DigitalSignerApplet' (GeneXus) and 'GXWS-Launcher' (GeneXus). Below this, 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. The 'Client side validation behaviour' section is highlighted with an orange arrow, and its properties are listed in the table below.

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

Below the 'Client side validation behaviour' section, the 'Defaults' section is visible, showing properties like 'Report output' (Only To File) and 'Exposed namespace' (TravelAgency).

Até aqui temos visto o comportamento padrão destas regras. Mas é interessante saber que temos um grupo de propriedades chamado **Client Side Validation Behaviour** que encontramos no nível da versão, que nos permite personalizar o comportamento e as mensagens das regras de muitas formas, tornando a aplicação mais atraente para usuários finais.

## Stop on error property

Customer

« < > » SELECT

Id

Name

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

Quando utilizamos a regra Error, vemos que embora a transação não permita gravar, permite passar para o próximo campo, depois de ter mostrado a mensagem de erro. Se nos interessar que em caso de ser disparado um erro, se mantenha o foco no controle e obrigue o usuário a corrigir o valor para poder passar ao campo seguinte, devemos mudar o valor da propriedade **Stop on error** para **Yes**.

## Validation message position property

The screenshot shows a 'Customer' form with fields for Id, Name, Last Name, Address, Phone, Email, and Added Date. A validation message 'Enter the customer name' is displayed below the Name field. A table titled 'Client side validation behaviour' is overlaid on the right, with an arrow pointing to the 'Validation message position' property, which is set to 'Bottom'.

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

Também poderemos definir o lugar onde vai a mensagem em relação ao campo, utilizando a propriedade *Validation Message Position*, que conta com os valores *Right* (que é o valor padrão), *Left*, *Top* e *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

No caso da mensagem se mostrar sobreposta a algum outro controle, deve ser configurada também a propriedade *Validation message overlap adjacent controls*, que controla se as mensagens devem ou não sobrepor-se.

## 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

A última propriedade do grupo é *Validation message display*, que conta com os valores *One at a time* (que é o valor padrão) e cujo objetivo é mostrar sempre a última mensagem de validação que ativa a aplicação.

## Validation message display property

Customer

« < > » SELECT

Id

Name  Enter the customer name

Last Name

Address

Phone  The phone is empty

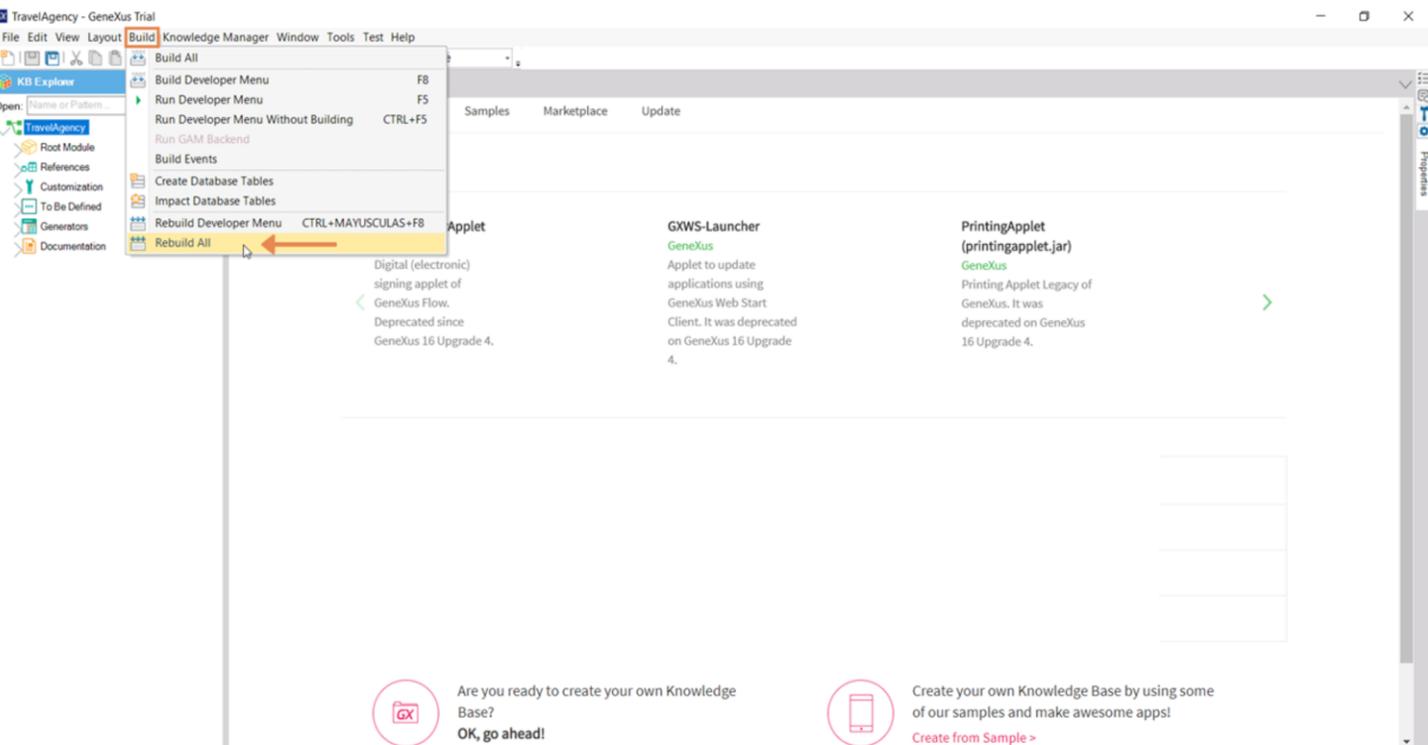
Email

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	All at once

e *All at once*, que mostra ao mesmo tempo cada mensagem de validação que deveria estar na tela.



Como estas propriedades modificam o comportamento relativo à interação nos forms, devem ser gerados novamente todos os objetos que têm form para que as mudanças sejam efetuadas. Para isso, deve utilizar a opção **Rebuild All**, que gera absolutamente todos os objetos.

No vídeo seguinte, adicionaremos algumas regras interessantes àquelas já conhecidas e em outros estudaremos como são avaliadas para determinar a ordem de execução, bem como nos aprofundaremos um pouco mais nos eventos de disparo.

# GeneXus™

[training.genexus.com](http://training.genexus.com)

[wiki.genexus.com](http://wiki.genexus.com)

[training.genexus.com/certifications](http://training.genexus.com/certifications)