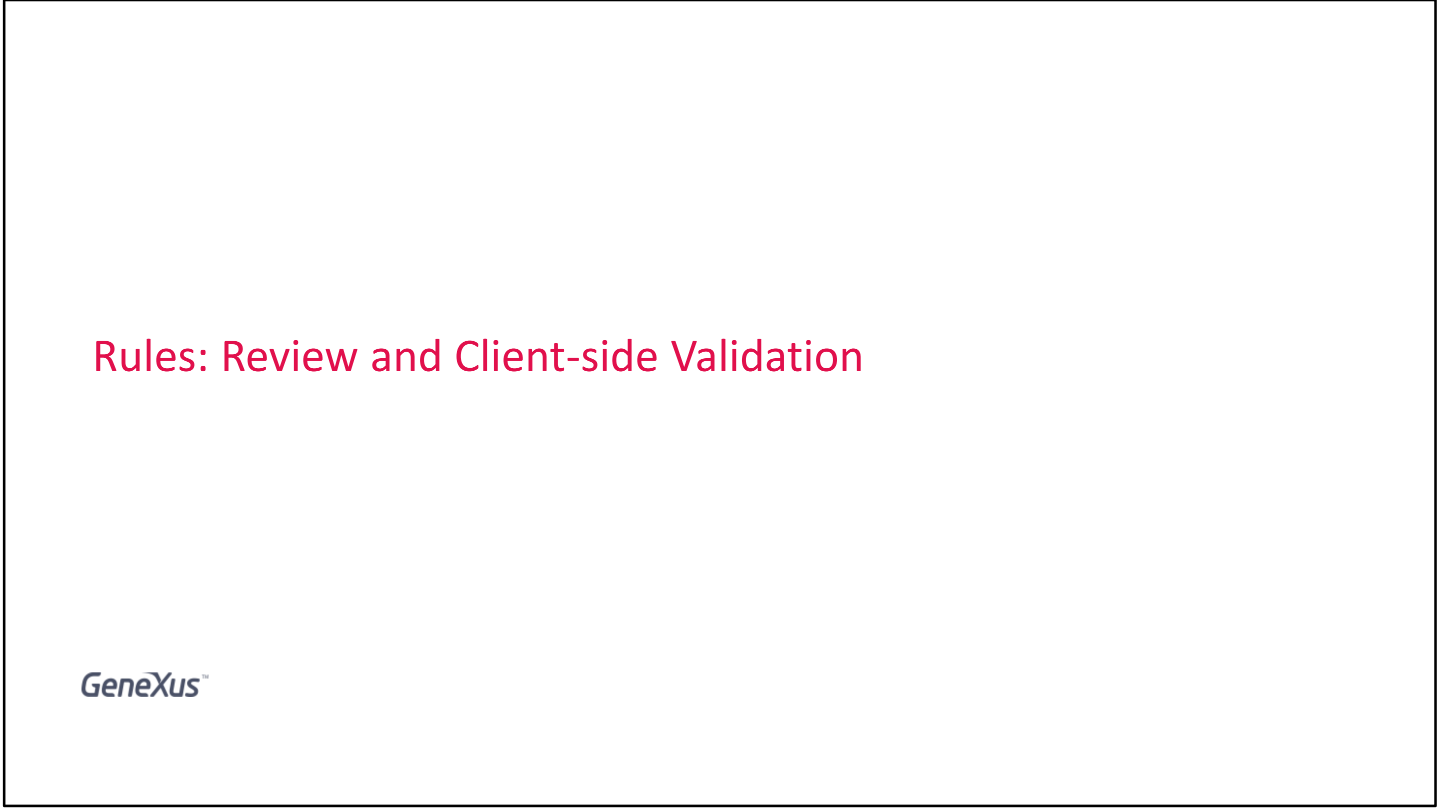
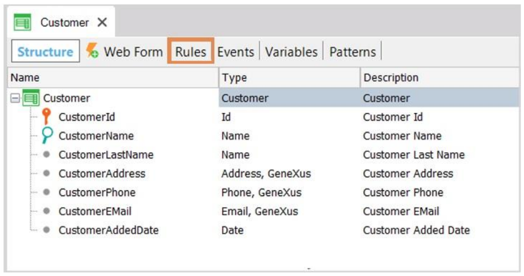
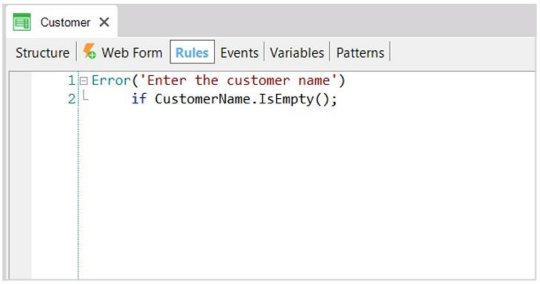


Rules: Review and Client-side Validation

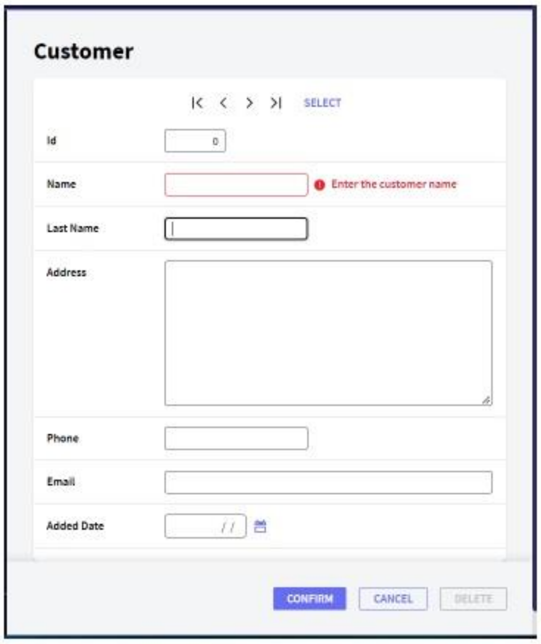
Rules



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



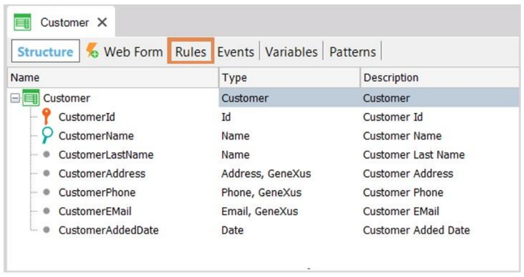
```
Error('Enter the customer name')
      if CustomerName.IsEmpty();
```



In the previous course we learned that the Transaction object has a section called **Rules**, in which we define the controls that must be made or the rules that must be fulfilled for a certain reality.

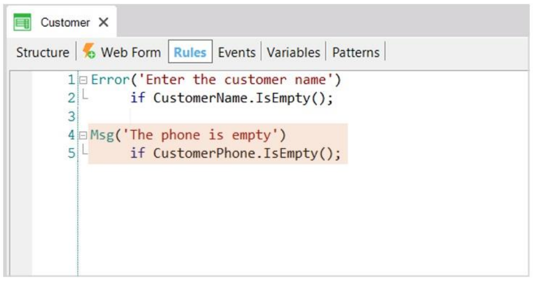
We focused on the **Error** rule, which prevents a record from being stored in the database while a certain condition is met: for example, if we are entering a client and leave his or her name empty, that client cannot be inserted until the user has typed in a name.

Rules


Customer

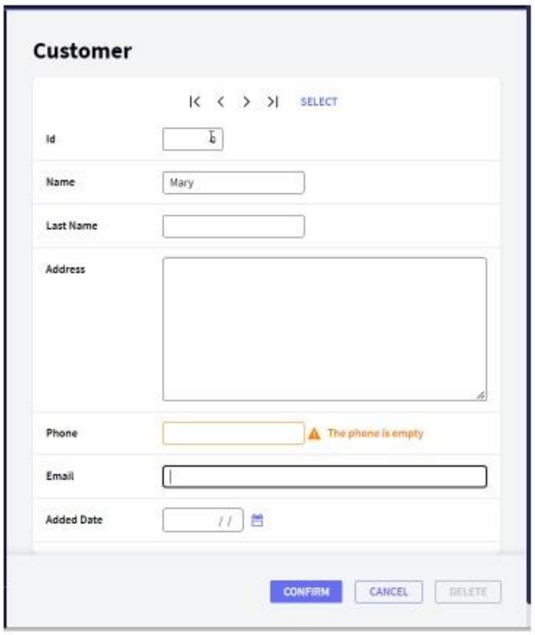
Structure | Web Form | Rules | Events | Variables | Patterns

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


Customer

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

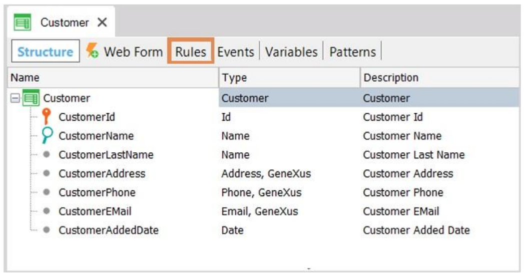
```
Error('Enter the customer name')
    if CustomerName.IsEmpty();

Msg('The phone is empty')
    if CustomerPhone.IsEmpty();
```



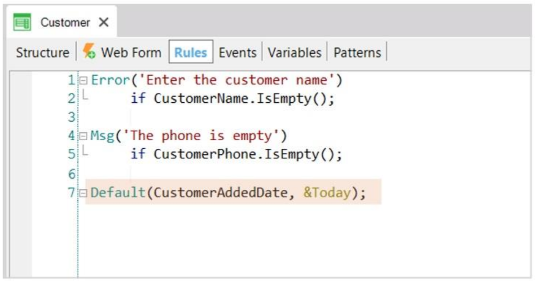
Id	
Name	Mary
Last Name	
Address	
Phone	The phone is empty
Email	
Added Date	/ /

We also saw the **Message** rule, which only informs the user through a message but allows saving;

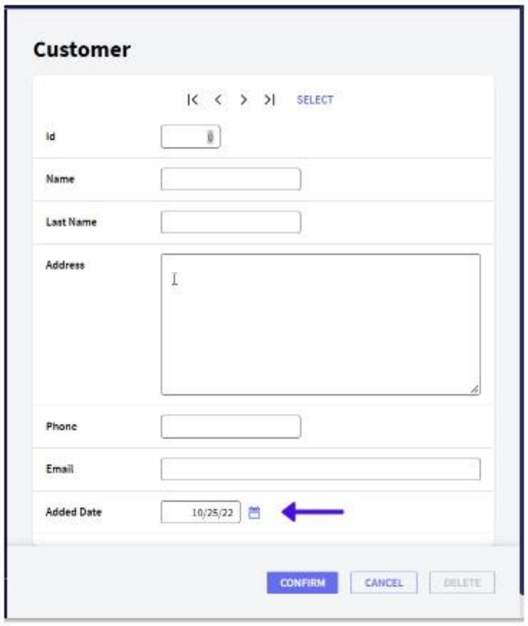
Rules



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



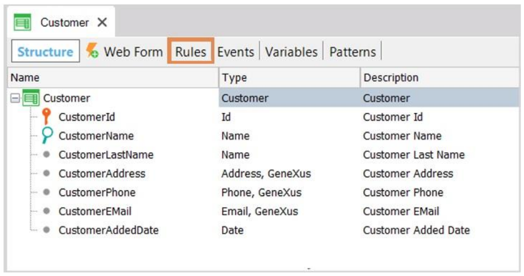
```
Error('Enter the customer name')
      if CustomerName.IsEmpty();

Msg('The phone is empty')
      if CustomerPhone.IsEmpty();

Default(CustomerAddedDate, &Today);
```



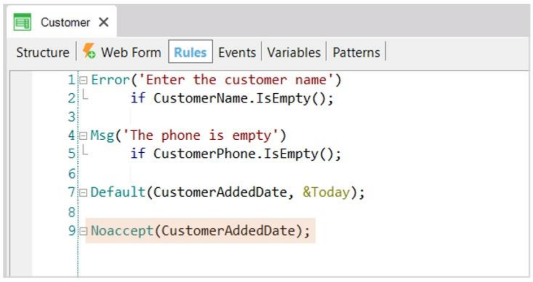
Id	0
Name	
Last Name	
Address	
Phone	
Email	
Added Date	10/25/22

the **Default** rule, which allows us to initialize an attribute or variable with a value when we access the transaction in insert mode;

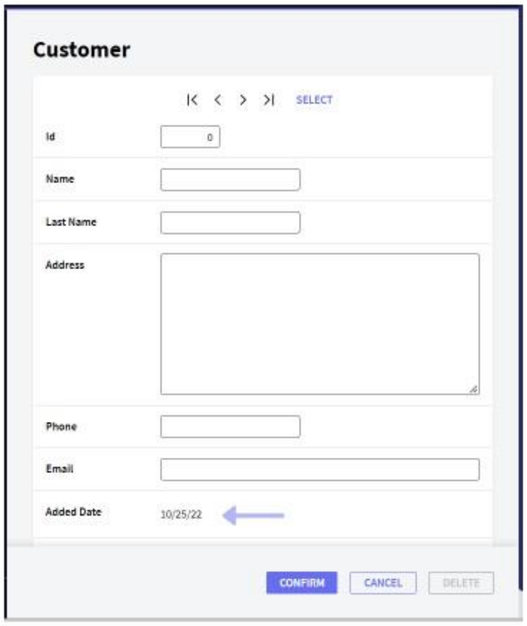
Rules



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



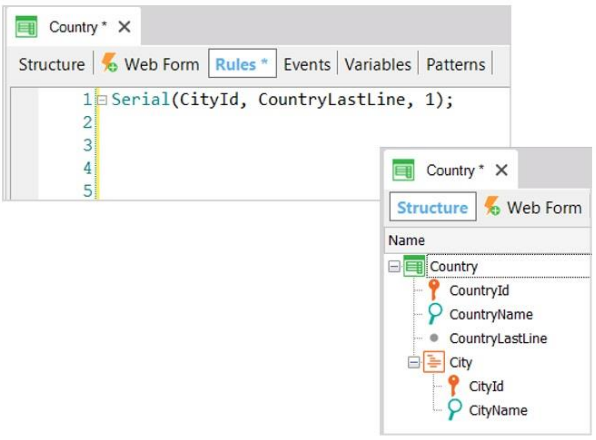
```
Error('Enter the customer name')
      if CustomerName.IsEmpty();

Msg('The phone is empty')
      if CustomerPhone.IsEmpty();

Default(CustomerAddedDate, &Today);

Noaccept(CustomerAddedDate);
```



Id	0
Name	
Last Name	
Address	
Phone	
Email	
Added Date	10/25/22

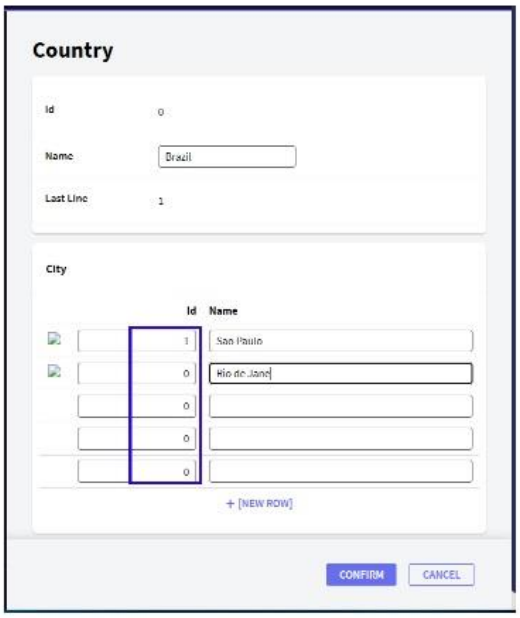
the **NoAccept** rule, which prevents the user from making changes to a field in the form: it shows it disabled;

Rules



```
Serial(CityId, CountryLastLine, 1);
```

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



Country

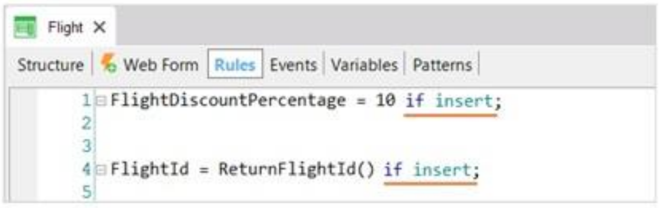
Id	0
Name	Brazil
Last Line	1

City

Id	Name
1	Sao Paulo
0	Rio de Jane
0	
0	
0	

and finally the **Serial** rule, which is used to auto-number a second, third, or other nested level of a transaction.

Rules



```
FlightDiscountPercentage = 10 if insert;


FlightId = ReturnFlightId() if insert;

```

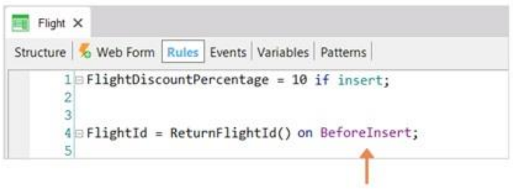
if insert

if update

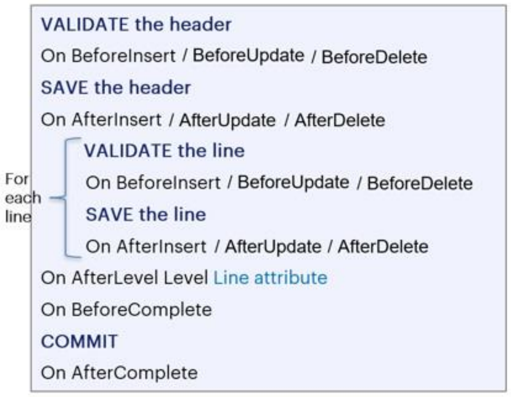
if delete

We also learned that through the rules we can define value assignments or invoke objects, and that we can also condition them to be executed only when we're making additions, modifications or deletions.

Rules

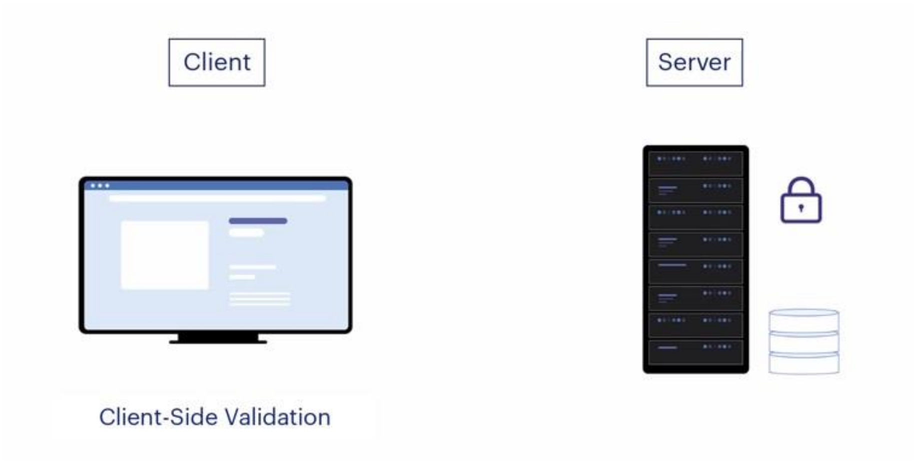


```
FlightDiscountPercentage = 10 if insert;


FlightId = ReturnFlightId() on BeforeInsert;
```


VALIDATE the header
On BeforeInsert / BeforeUpdate / BeforeDelete
SAVE the header
On AfterInsert / AfterUpdate / AfterDelete
For each line
VALIDATE the line
On BeforeInsert / BeforeUpdate / BeforeDelete
SAVE the line
On AfterInsert / AfterUpdate / AfterDelete
On AfterLevel Level Line attribute
On BeforeComplete
COMMIT
On AfterComplete

In addition, if the moment chosen by GeneXus to execute a rule is not the one we need, we can indicate the exact moment we want it to be executed, using triggering events.

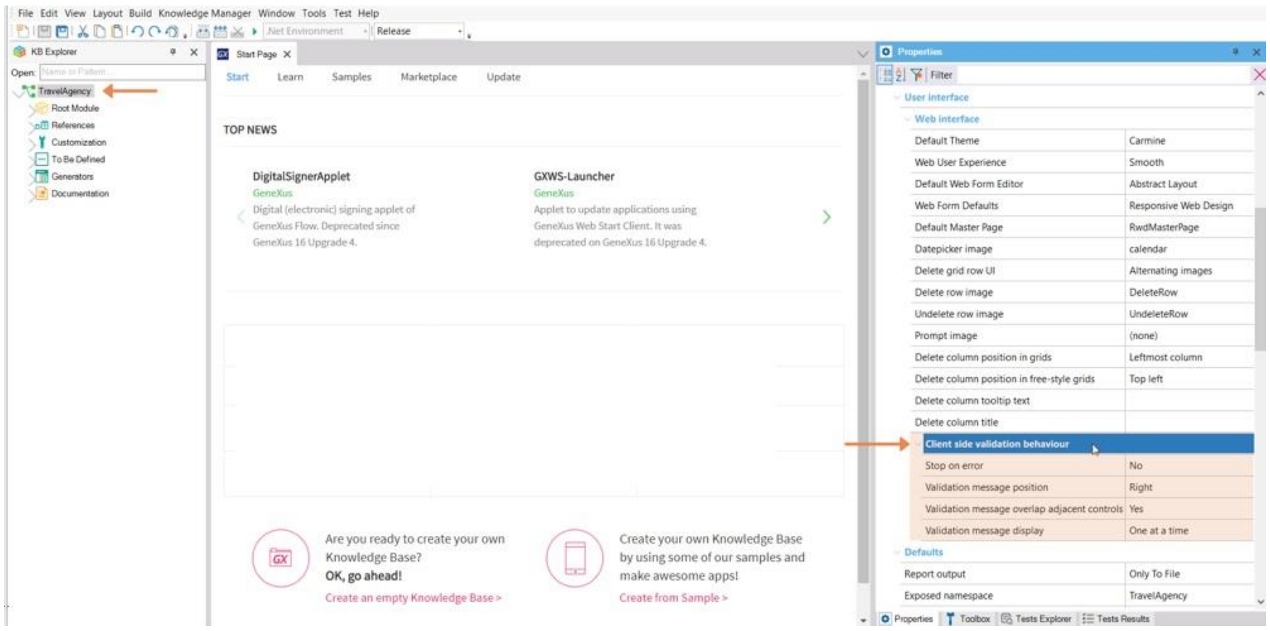
Execution of rules



Remember that all these rules we have studied are validated both in the web client and in the server.

The validation that occurs in the client is called **Client-side Validation,** and its purpose is to provide a good user experience, making the user feel that the application is interacting with him all the time. But actually the server validates that all the information sent is consistent and does not breach the security of the system, and it is the only one that can operate on the database.

Client side validation



TOP NEWS

DigitalSignerApplet
GeneXus
Digital (electronic) signing applet of GeneXus Flow. Deprecated since GeneXus 16 Upgrade 4.

GXWS-Launcher
GeneXus
Applet to update applications using GeneXus Web Start Client. It was deprecated on GeneXus 16 Upgrade 4.

Are you ready to create your own Knowledge Base?
OK, go ahead!
Create an empty Knowledge Base >

Create your own Knowledge Base by using some of our samples and make awesome apps!
Create from Sample >

User interface

Web interface

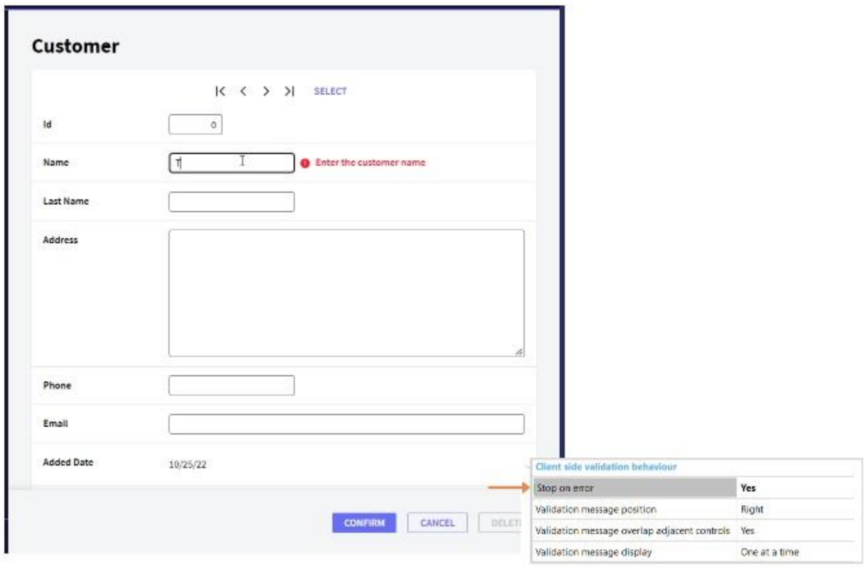
Default Theme	Carmine
Web User Experience	Smooth
Default Web Form Editor	Abstract Layout
Web Form Defaults	Responsive Web Design
Default Master Page	RwdMasterPage
Datepicker image	calendar
Delete grid row UI	Alternating images
Delete row image	DeleteRow
Undelete row image	UndeleteRow
Prompt image	(none)
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

So far we have seen the default behavior of these rules. But we need to know that there is a group of properties called Client Side Validation Behaviour available at the version level, which allows us to customize the behaviour and messages of the rules in many ways, making the application more attractive to end users.

Stop on error property



Customer

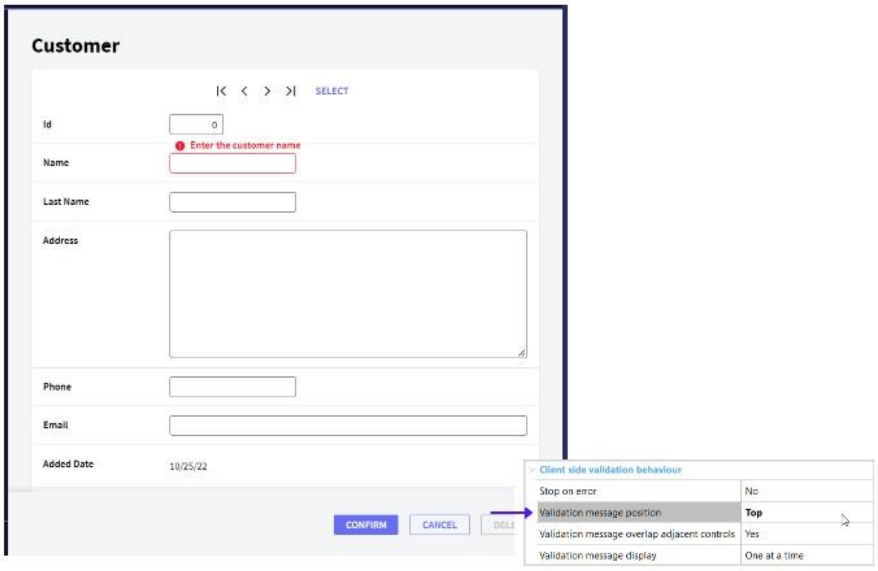
Id	0
Name	T — Enter the customer name
Last Name	
Address	
Phone	
Email	
Added Date	10/25/22

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

When we used the Error rule, we saw that although the transaction doesn't allow saving, it does allow moving to the next field after showing the error message. If we want to keep the focus on the control and require the user to correct the value in order to move to the next field when an error is triggered, we must change the value of the **Stop on error** property to Yes.

Validation message position property



Enter the customer name

0

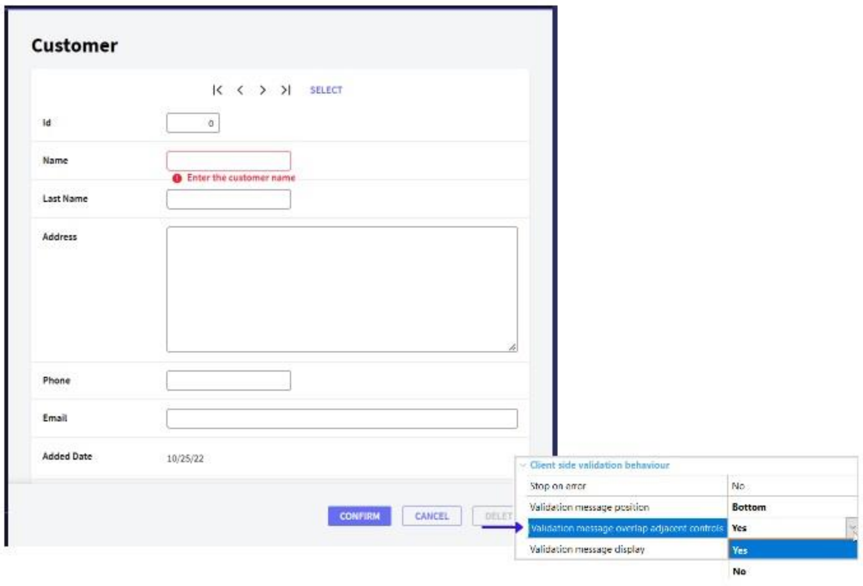
10/25/22

Client side validation behaviour

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

We can also define the position of the message in relation to the field, using the **Validation Message Position** property, which has the values Right (which is the default value), Left, Top and Bottom.

Validation message overlap adjacent property



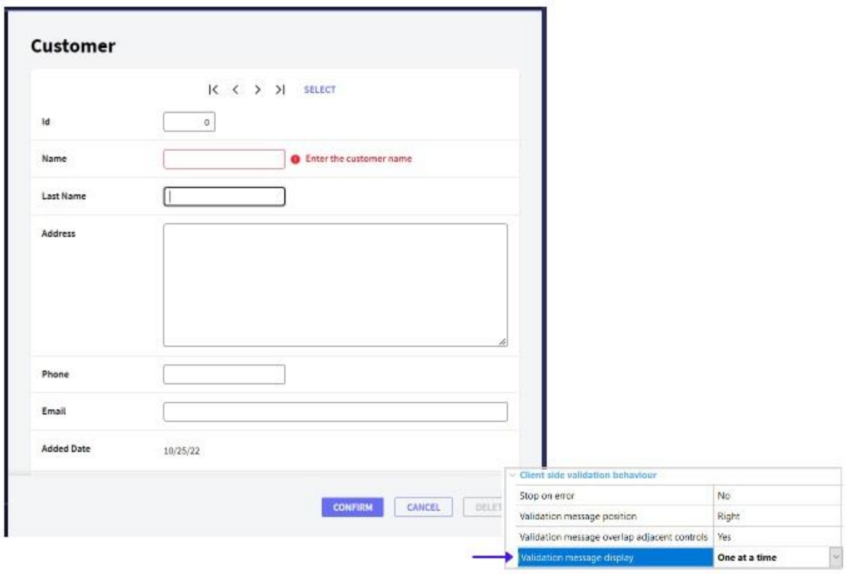
Client side validation behaviour

Stop on error	No
Validation message position	Bottom
Validation message overlap adjacent controls	Yes
Validation message display	

Yes
No

If the message is overlapped with another control, the **Validation message overlap adjacent** controls property must also be set, which controls whether the messages should overlap or not.

Validation message display property



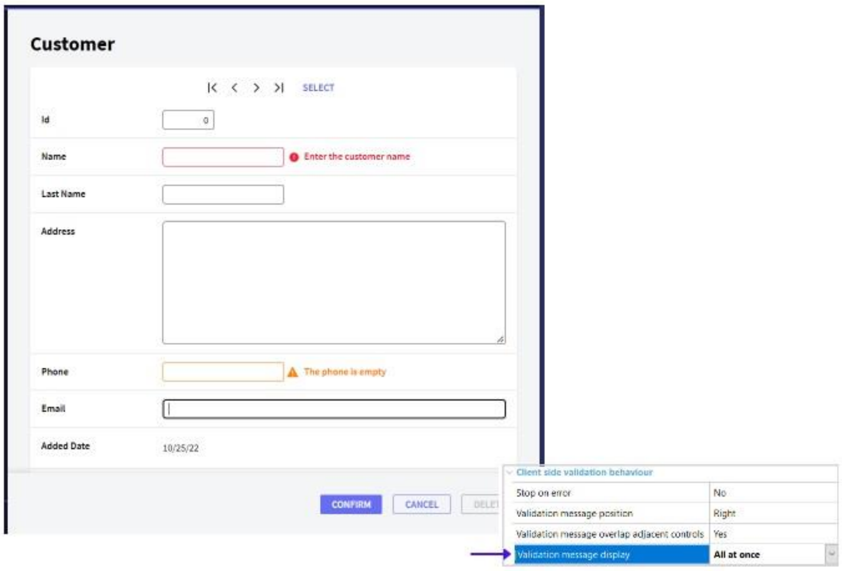
Customer

Id	0
Name	Enter the customer name
Last Name	
Address	
Phone	
Email	
Added Date	10/25/22

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

The last property of the group is **Validation message display**, which has the values One at a time (which is the default value) and whose objective is to always show the last validation message activated by the application,

Validation message display property



Customer

Id	0
Name	Enter the customer name
Last Name	
Address	
Phone	The phone is empty
Email	
Added Date	10/25/22

Client side validation behaviour

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

and All at once, which displays every validation message that should be on the screen at the same time.

Since these properties change the behavior related to the interaction in the forms, all the objects that have a form must be generated again for the changes to be made. To do this, you must use the **Rebuild All** option, which generates absolutely all objects.

In the following video we will add some interesting rules to those we already know, and in others we will explore how they are evaluated to determine the order of execution, as well as take a closer look at the triggering events.

training.genexus.com
wiki.genexus.com