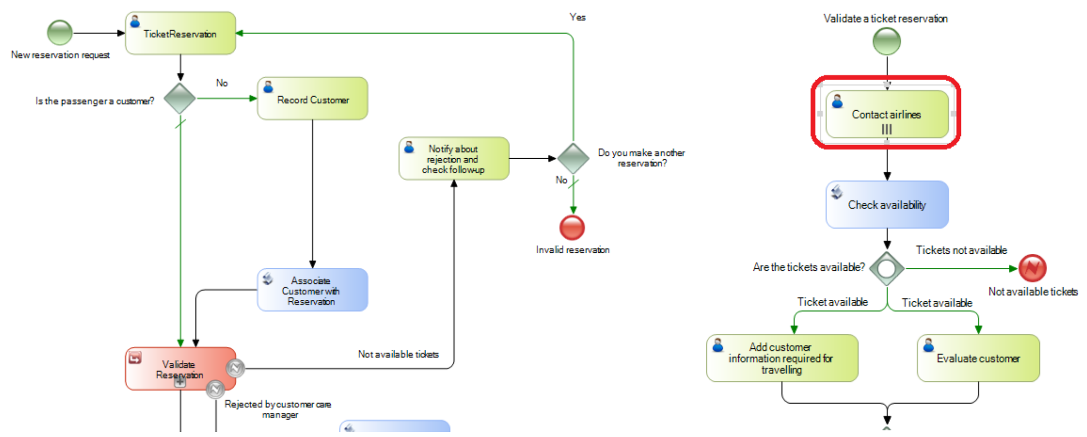
Starting a process from a GeneXus object using the Workflow API

The travel agency decided to modify its ticket reservation process and include some new functionalities.

First it requested the possibility of allowing agency customers to enter their own reservations through the website.

Customers must be logged on the site and provide the reservation data (date, departure airport, arrival airport, number of passengers, etc.), to allow the system to create a reservation and start the FlightTicketReservation process to later indicate the TicketReservation task as completed.

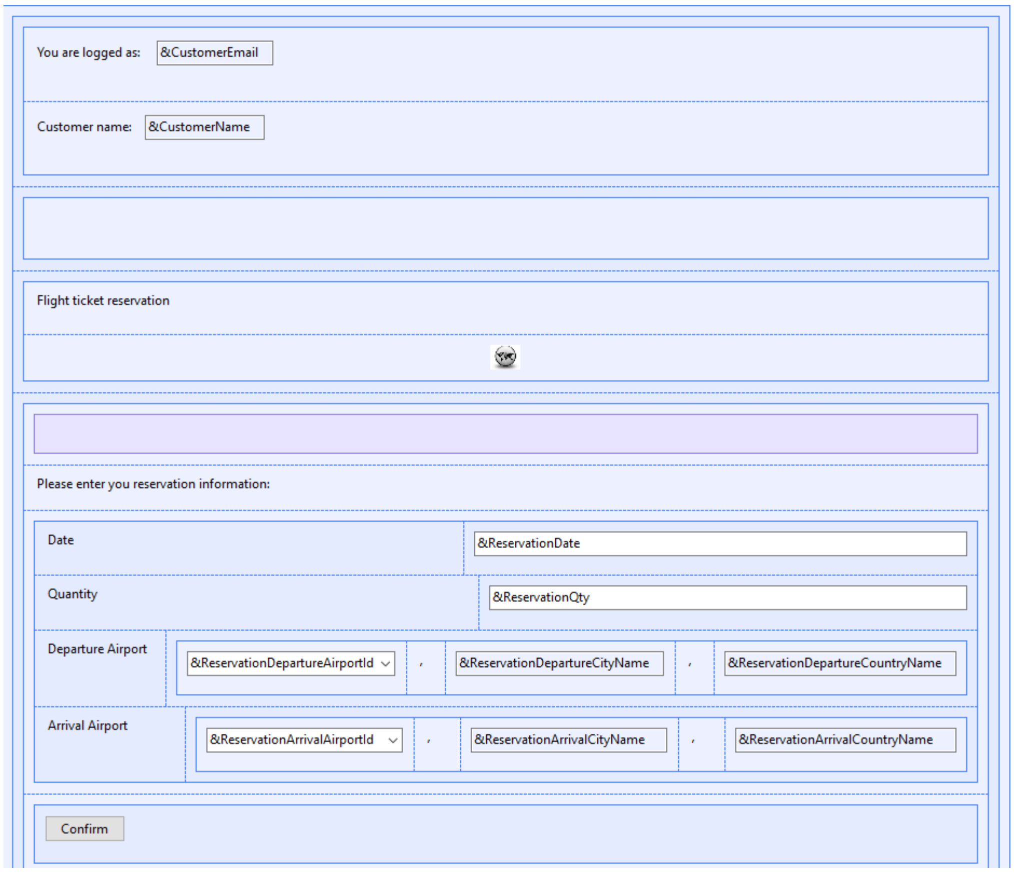
When the passenger is already a customer of the agency, once the TicketReservation task is completed, the following task to execute is the first task in the ValidateReservation subprocess: ContactAirlines, which in turn is a multi-instance task.


New reservation request
TicketReservation
Yes
Is the passenger a customer?
No
Record Customer
Notify about rejection and check follow-up
Do you make another reservation?
No
Invalid reservation
Associate Customer with Reservation
Validate Reservation
Not available tickets
Rejected by customer care manager
Validate a ticket reservation
Contact airlines
Check availability
Tickets not available
Are the tickets available?
Not available tickets
Ticket available
Ticket available
Add customer information required for travelling
Evaluate customer

The systems must also notify each person able to contact airlines that the instances of the ContactAirlines task are available to be executed. And lastly, substitute the task type notifying customers in regards to the reservation being authorized so that the notification is done in person.

Let’s start by the customer entering the reservation in the website.

To that end we will use a web panel called TravelAgency. This web panel includes variables on screen with which the user will enter the reservation data, in addition to a Confirm button.


You are logged as: &CustomerEmail

Customer name: &CustomerName

Flight ticket reservation

Please enter you reservation information:

Date	&ReservationDate
Quantity	&ReservationQty
Departure Airport	&ReservationDepartureAirportId , &ReservationDepartureCityName , &ReservationDepartureCountryName
Arrival Airport	&ReservationArrivalAirportId , &ReservationArrivalCityName , &ReservationArrivalCountryName

For simplifying purposes in this demo, we will assume that the customer entering is Customer 1 who is already logged in, so we will not include the login controls. If we go to the Start event we will see the code to simulate this.

```
Event Start
    &CustomerId = 1 // Hardcoded only for demo uses
    &CustomerEmail  = GetLoggedUser(&CustomerId)
    &CustomerName  = GetCustomerName(&CustomerId)
Endevent
```

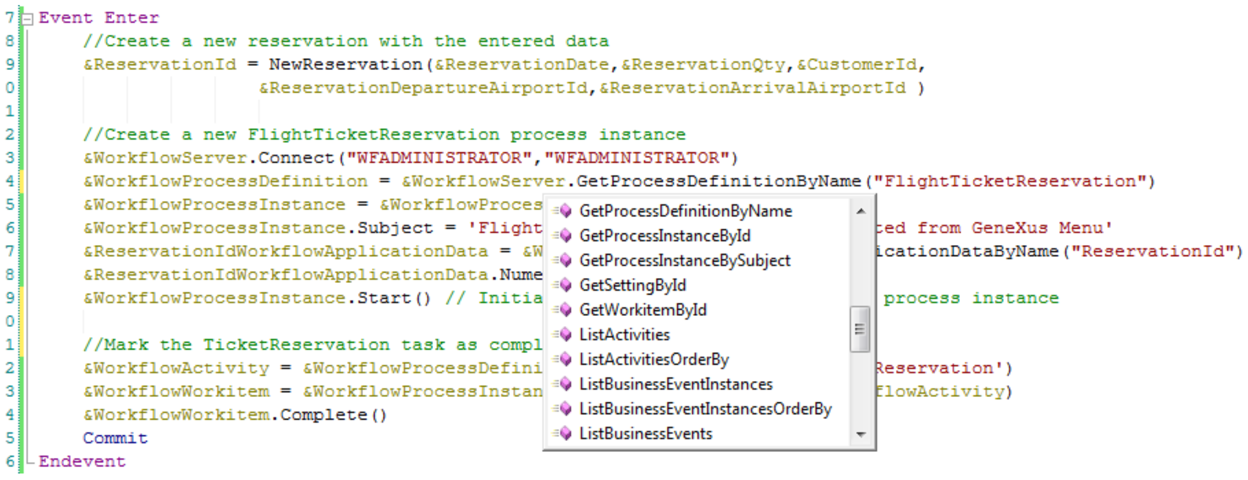
If we press the Confirm button the Enter event will be executed and several tasks will be done.

```
Event Enter
    //Create a new reservation with the entered data
    &ReservationId = NewReservation(&ReservationDate,&ReservationQty,&CustomerId,
                    &ReservationDepartureAirportId,&ReservationArrivalAirportId )

    //Create a new FlightTicketReservation process instance
    &WorkflowServer.Connect("WFADMINISTRATOR","WFADMINISTRATOR")
    &WorkflowProcessDefinition = &WorkflowServer.GetProcessDefinitionByName("FlightTicketReservation")
    &WorkflowProcessInstance = &WorkflowProcessDefinition.CreateInstance()
    &WorkflowProcessInstance.Subject = 'FlightTicketReservation process started from GeneXus Menu'
    &ReservationIdWorkflowApplicationData = &WorkflowProcessInstance.GetApplicationDataByName("ReservationId")
    &ReservationIdWorkflowApplicationData.NumericValue = &ReservationId
    &WorkflowProcessInstance.Start() // Initiate the FlightTicketReservation process instance

    //Mark the TicketReservation task as completed
    &WorkflowActivity = &WorkflowProcessDefinition.GetActivityByName('TicketReservation')
    &WorkflowWorkitem = &WorkflowProcessInstance.GetWorkitemByActivity(&WorkflowActivity)
    &WorkflowWorkitem.Complete()
    Commit
Endevent
```

The NewReservation method is invoked first to create a reservation in the database. Then, using **Workflow data types**, the FlightTicketReservation process is initiated and the TicketReservation task is marked as completed.

These data types starting by the prefix Workflow are GeneXus data types which allow for the application to interact with the Workflow engine.

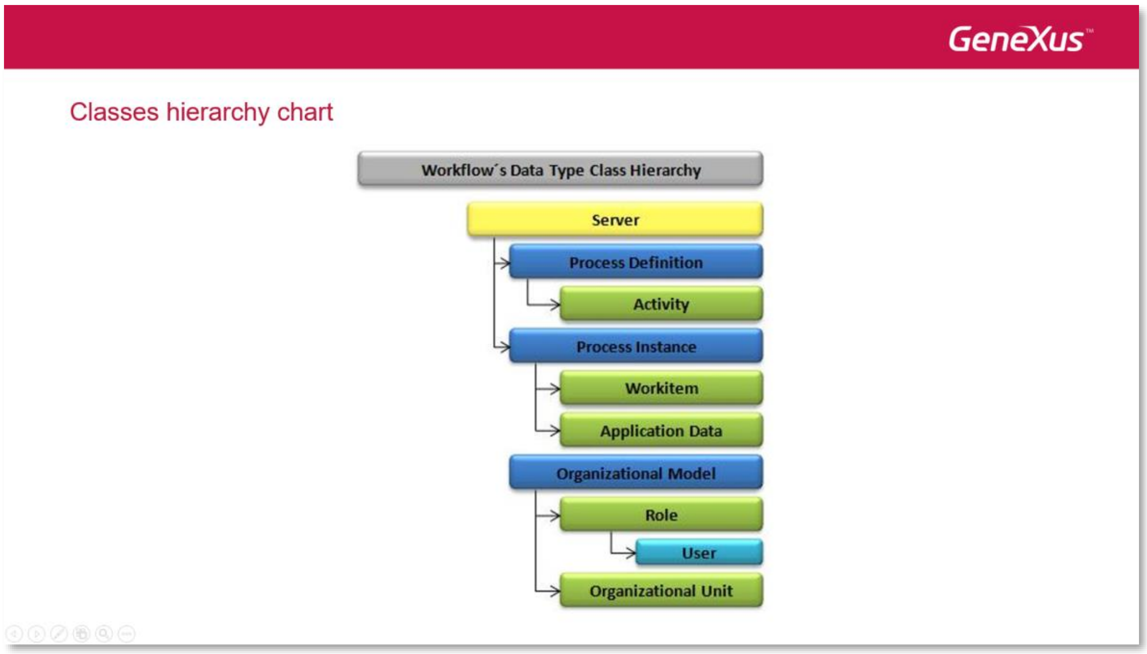
In order to use a Workflow data type and interact with the engine's API we must first define a variable with that data type. Then, with the help of the context information we can choose the method or property we want to use.



```
Event Enter
    //Create a new reservation with the entered data
    &ReservationId = NewReservation(&ReservationDate,&ReservationQty,&CustomerId,
                    &ReservationDepartureAirportId,&ReservationArrivalAirportId )

    //Create a new FlightTicketReservation process instance
    &WorkflowServer.Connect("WFADMINISTRATOR","WFADMINISTRATOR")
    &WorkflowProcessDefinition = &WorkflowServer.GetProcessDefinitionByName("FlightTicketReservation")
    &WorkflowProcessInstance = &WorkflowProces
    &WorkflowProcessInstance.Subject = 'Flight                    ted from GeneXus Menu'
    &ReservationIdWorkflowApplicationData = &W                    icationDataByName("ReservationId")
    &ReservationIdWorkflowApplicationData.Nume
    &WorkflowProcessInstance.Start() // Initia                     process instance

    //Mark the TicketReservation task as compl
    &WorkflowActivity = &WorkflowProcessDefini                    Reservation')
    &WorkflowWorkitem = &WorkflowProcessInstan                    flowActivity)
    &WorkflowWorkitem.Complete()
    Commit
Endevent
```

GetProcessDefinitionByName
GetProcessInstanceById
GetProcessInstanceBySubject
GetSettingById
GetWorkitemById
ListActivities
ListActivitiesOrderBy
ListBusinessEventInstances
ListBusinessEventInstancesOrderBy
ListBusinessEvents

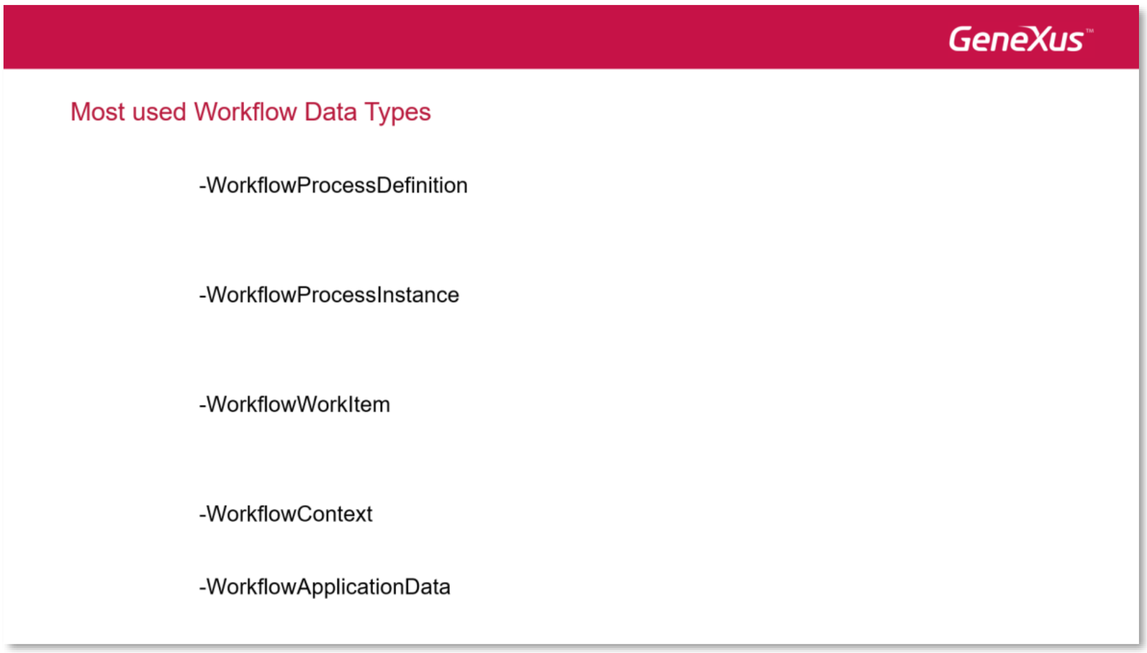
Workflow data types are classified into a hierarchy of classes.



The highest class is the **Server** class on which 3 other classes depend, namely: **Process Definition**, which allows us to access the components of a process diagram, **Process Instance**, which allows us to access an instance of a process under execution, and **Organization Model**, which allows us to access the information regarding the company's organizational structure, such as roles and users.

The Server class is the entry point of the types hierarchy and its methods enable us to access any workflow data type.

The data types most frequently used are:



GeneXus™

Most used Workflow Data Types

-WorkflowProcessDefinition

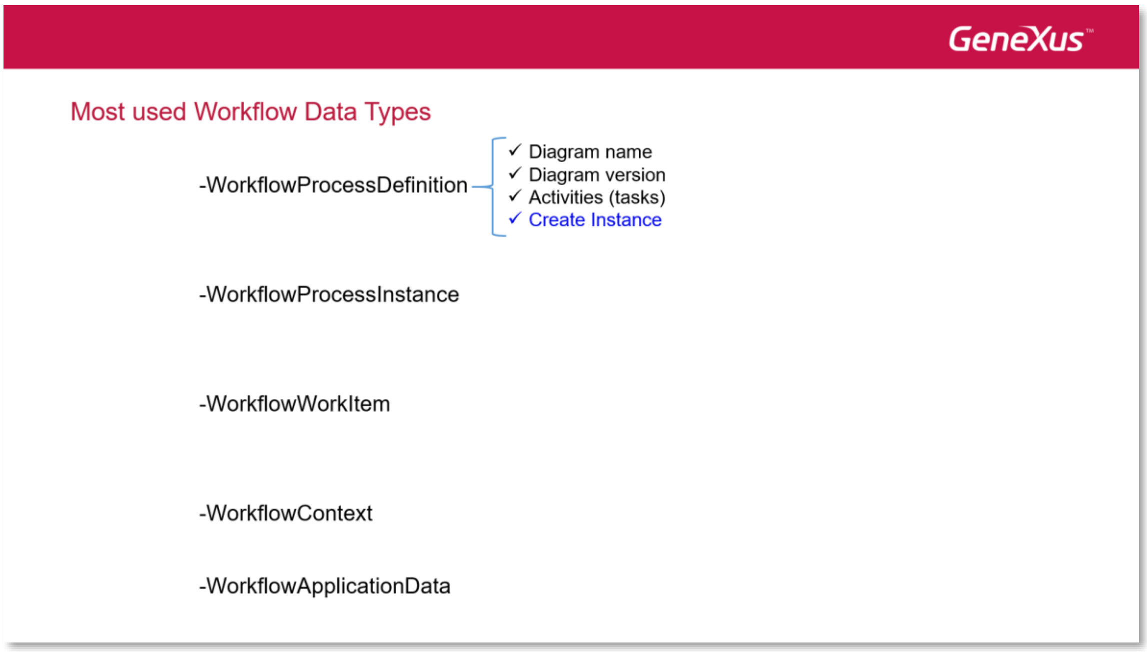
-WorkflowProcessInstance

-WorkflowWorkItem

-WorkflowContext

-WorkflowApplicationData

With **WorkflowProcessDefinition** we can access several properties of the diagram, such as name, version, tasks including it (which we call activities) and we can also create an execution instance of the process with the CreateInstance method, based on that definition.



GeneXus™

Most used Workflow Data Types

-WorkflowProcessDefinition
- ✓ Diagram name
- ✓ Diagram version
- ✓ Activities (tasks)
- ✓ Create Instance

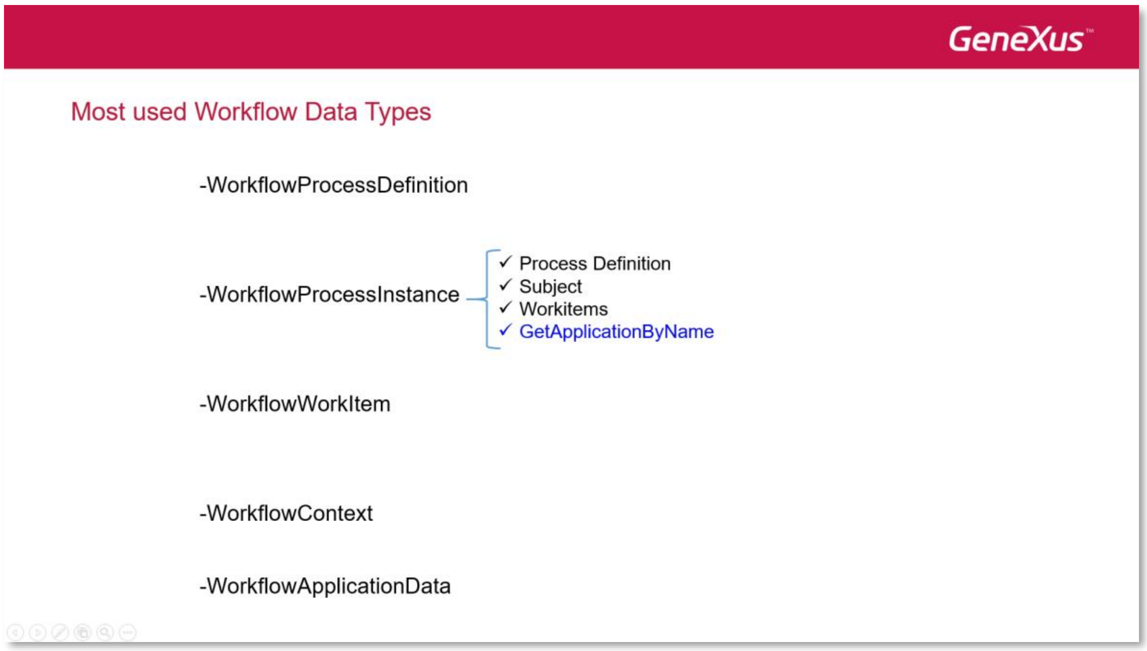
-WorkflowProcessInstance

-WorkflowWorkItem

-WorkflowContext

-WorkflowApplicationData

Using WorkflowProcessInstance we can find the definition of the process on which the instance is based, the issue that the instance deals with, and the collection of workitems that are part of the instance. Through the GetApplicationByName method we can retrieve relevant data by using its name.



GeneXus™

Most used Workflow Data Types

-WorkflowProcessDefinition

-WorkflowProcessInstance
- ✓ Process Definition
- ✓ Subject
- ✓ Workitems
- ✓ GetApplicationByName

-WorkflowWorkItem

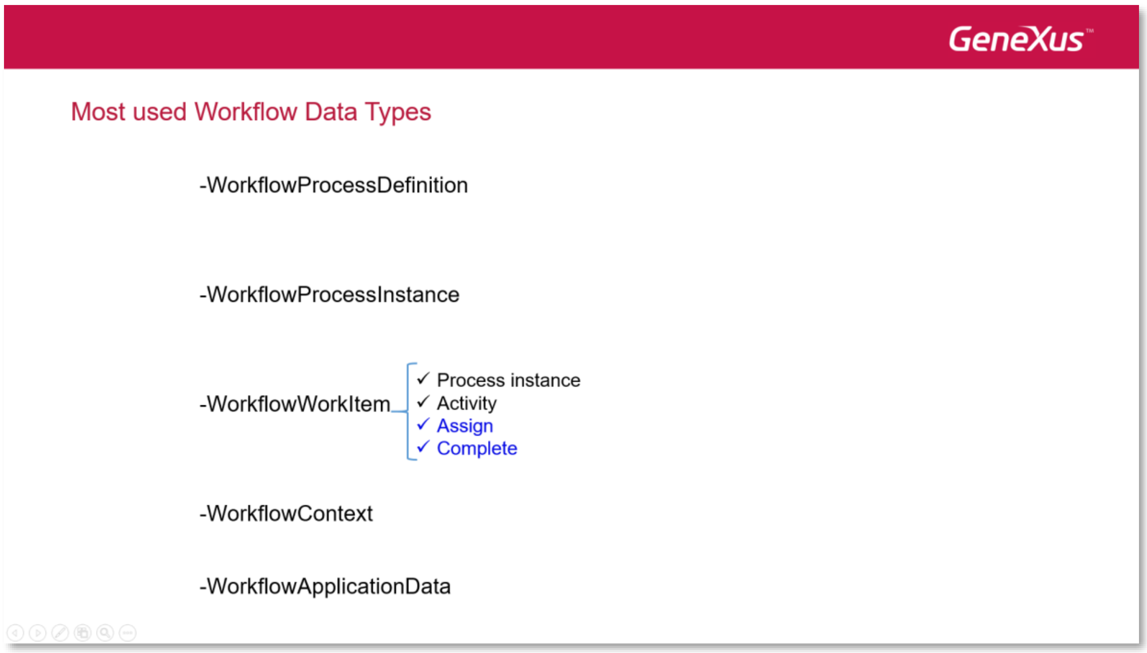
-WorkflowContext

-WorkflowApplicationData

The WorkflowWorkItem class enables us to know the work that needs to be done by participants in the context of an activity, within the process instance.

Its ProcessInstance provides us with information on the process instance to which the workitem belongs, and the Activity property return the activity that generated the workitem.

The Assign method enables us to assign a workitem to a specific user and the Complete method enables us to end the execution of the workitem.



GeneXus™

Most used Workflow Data Types

-WorkflowProcessDefinition

-WorkflowProcessInstance

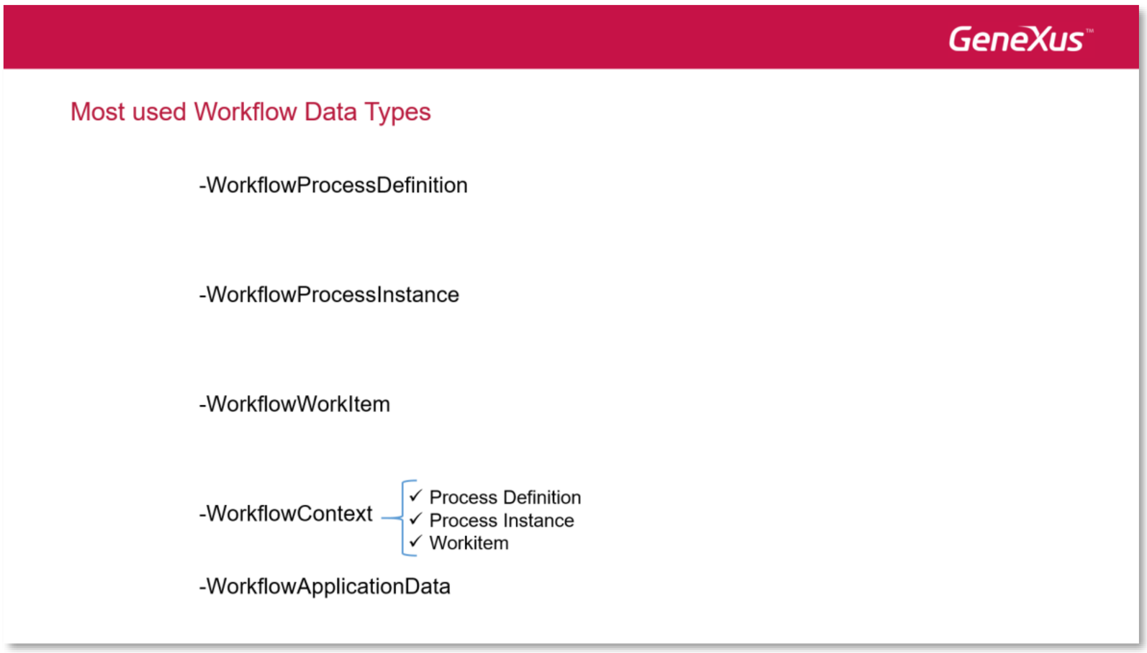
-WorkflowWorkItem
- ✓ Process instance
- ✓ Activity
- ✓ Assign
- ✓ Complete

-WorkflowContext

-WorkflowApplicationData

The WorkflowContext data type provides us with information on the context of execution of an application associated with an activity. This context is automatically instanced when the application associated with the activity (that is, the task) is a GeneXus object that is part of the same KB that contains the process diagram.

This automatic instancing of the context enables us to know the values of the process definition, the instance of the process and the workitem associated with the activity.



Most used Workflow Data Types

-WorkflowProcessDefinition

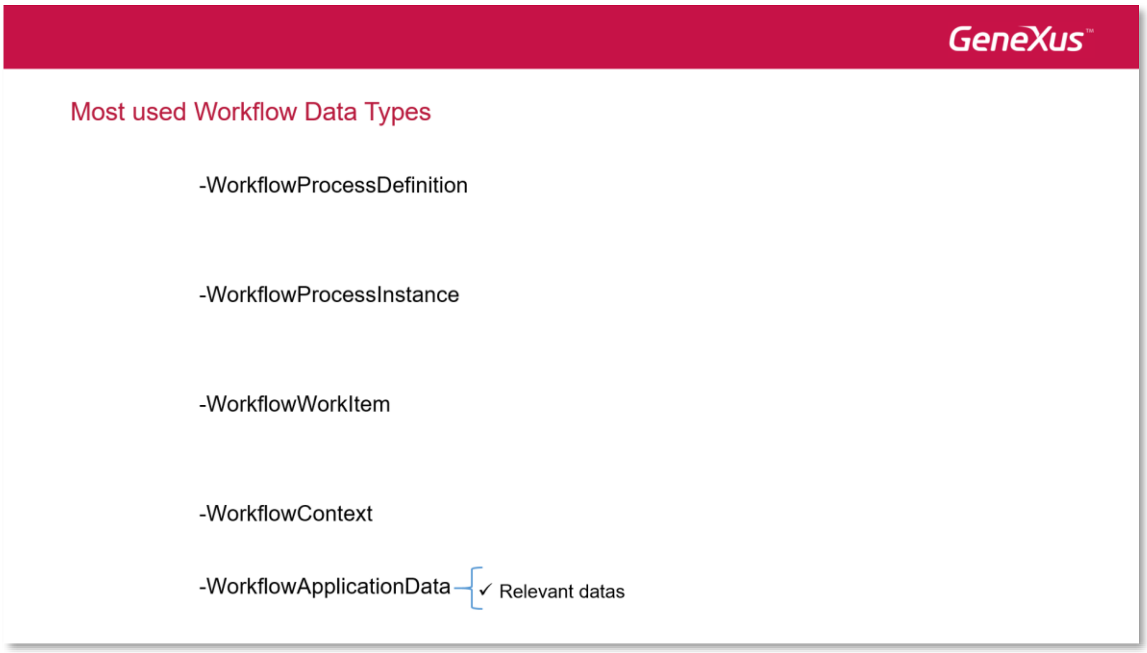
-WorkflowProcessInstance

-WorkflowWorkItem

-WorkflowContext
- ✓ Process Definition
- ✓ Process Instance
- ✓ Workitem

-WorkflowApplicationData

Lastly, the WorkflowApplicationData data type is the one we use when we want to work with relevant data, like when we store relevant data that we obtain through the GetApplicationDataByName method.



Most used Workflow Data Types

-WorkflowProcessDefinition

-WorkflowProcessInstance

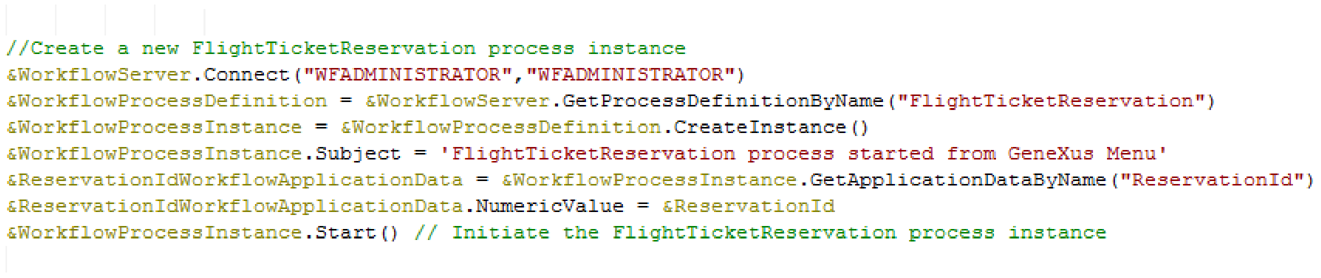
-WorkflowWorkItem

-WorkflowContext

-WorkflowApplicationData
- ✓ Relevant datas

Now back at the event of the webpanel that invokes the FlightTicketReservation process, we have here a defined variable, &WorkflowServer, of the WorkflowServer type. In practice we always use names of variables matching the Workflow data types to make it easier to identify them.

The first operation we perform with the WorkflowServer data type is to connect to the workflow engine using the administrator password.



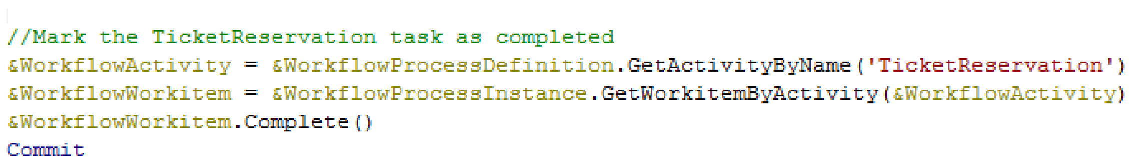
```
//Create a new FlightTicketReservation process instance
&WorkflowServer.Connect("WFADMINISTRATOR","WFADMINISTRATOR")
&WorkflowProcessDefinition = &WorkflowServer.GetProcessDefinitionByName("FlightTicketReservation")
&WorkflowProcessInstance = &WorkflowProcessDefinition.CreateInstance()
&WorkflowProcessInstance.Subject = 'FlightTicketReservation process started from GeneXus Menu'
&ReservationIdWorkflowApplicationData = &WorkflowProcessInstance.GetApplicationDataByName("ReservationId")
&ReservationIdWorkflowApplicationData.NumericValue = &ReservationId
&WorkflowProcessInstance.Start() // Initiate the FlightTicketReservation process instance
```

We then obtain the definition of the FlightTicketReservation process based on its name and save it in a variable of the WorkflowProcessDefinition type.

Once we have the definition we create an instance in the process with the CreateInstance method. Then we change subjects so as to recognize the process easily in the input tray.

Afterwards, we load relevant data ReservationId with the reservation identifier we created previously and then start the instance with the Start() method.

The following code lines are used to mark the TicketReservation task as completed.



```
//Mark the TicketReservation task as completed
&WorkflowActivity = &WorkflowProcessDefinition.GetActivityByName('TicketReservation')
&WorkflowWorkitem = &WorkflowProcessInstance.GetWorkitemByActivity(&WorkflowActivity)
&WorkflowWorkitem.Complete()
Commit
```

First we obtain the TicketReservation activity from the definition of the process, and with that activity we obtain the workitem corresponding to the task under execution in the process. Then we mark the workitem (the task) as completed.

We should note that following the Complete() method there is a Commit. The changes made using the workflow data types are included within the application's **Logic Work Unit**.

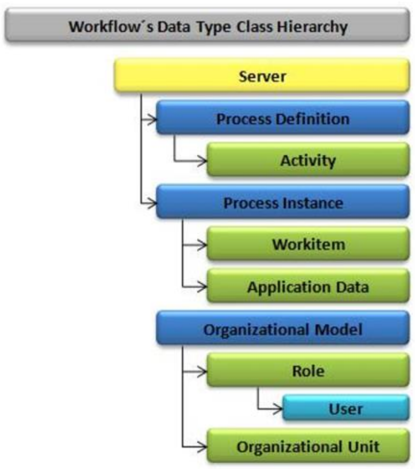
However, **workflow operations to not perform Commit**, so we must make sure that we define the UTL in the application correctly and do the Commit in the end. In this case, we did the workflow operations on a webpanel, so we will need to add the Commit when we finish them.

These workflow data types we saw are a subgroup of all the ones available, and we can do many tasks by code, through the API of the workflow engine.

Further information on this topic is available at the following link.

GeneXus™

Classes hierarchy chart



http://wiki.gxtechnical.com/commwiki/servlet/hwiki?Category%3AWorkflow+Data+Types,