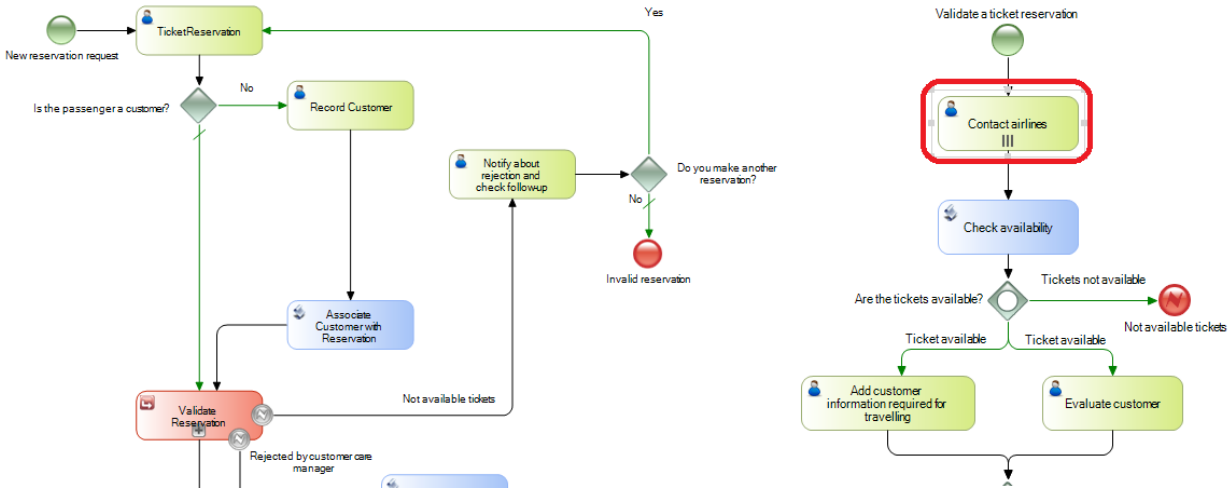
Inicio de un proceso desde un objeto GeneXus, usando la API de Workflow

La agencia de viajes decidió modificar su proceso de reserva de pasajes e incorporar algunas funcionalidades nuevas.

En primer lugar, solicitó que sea posible que un cliente de la agencia pueda ingresar su propia reserva a través del sitio web.

El cliente deberá estar logueado al sitio y proveer los datos de la reserva (fecha, aeropuerto de origen, aeropuerto de destino, cantidad de personas, etc.), con lo cual el sistema creará una reserva, iniciará el proceso FlightTicketReservation y marcará la tarea TicketReservation como completada.

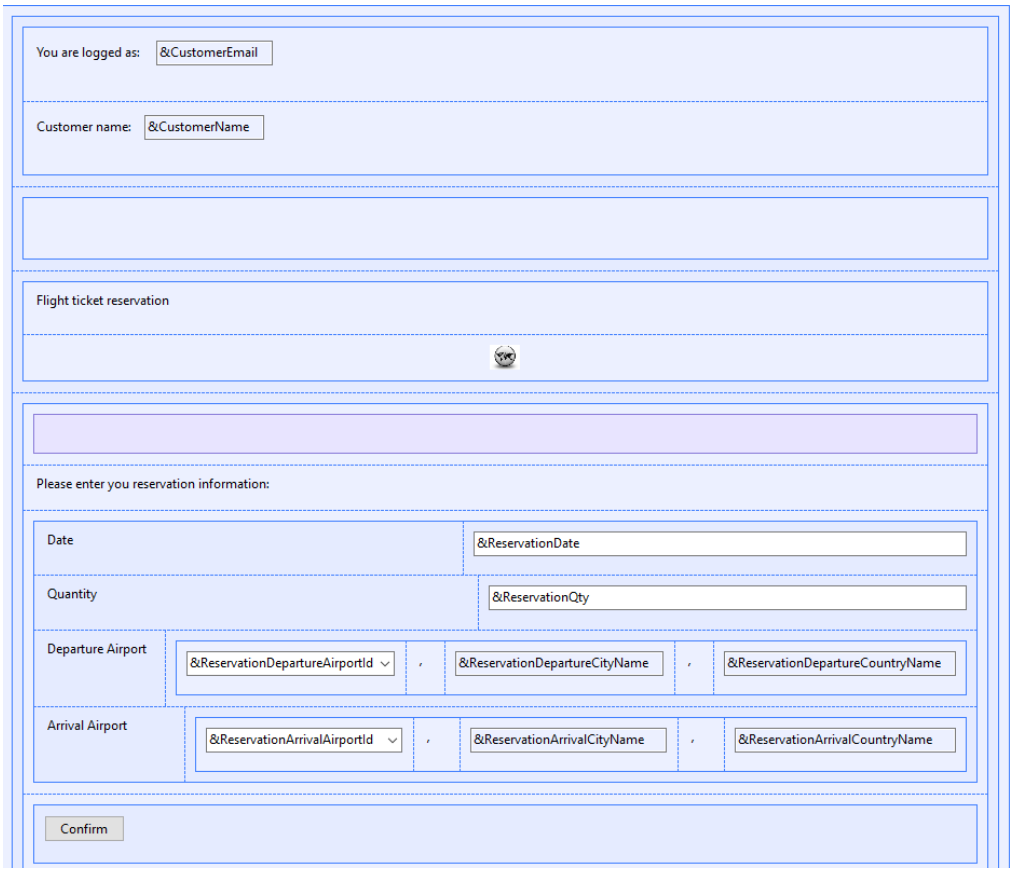
Como el pasajero ya es cliente de la empresa, una vez cumplida la tarea TicketReservation, la siguiente tarea pendiente de ejecución es la primera tarea del subproceso ValidateReservation, ContactAirlines, que es a su vez una tarea multi-instanciada.



El sistema además deberá avisarle por mail a cada una de las personas que pueden contactar aerolíneas, que las instancias de la tarea ContactAirlines están disponibles para ser ejecutadas. Y por último, sustituir el tipo de tarea que notificaba al cliente de que la reserva fue autorizada, de forma de que el aviso sea en persona.

Comencemos por el ingreso de la reserva por parte del cliente, en el sitio web.

Para eso utilizaremos un webpanel llamado TravelAgency. Este web panel tiene variables en pantalla mediante las cuales el usuario ingresará los datos de la reserva y un botón de confirmar.



Por razones de simplicidad para la demo, asumiremos que el cliente que ingresa es el cliente 1 y que ya está logueado, por lo cual no incluiremos los controles de login. Si vamos al evento Start, veremos el código para simular esto.

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

Al presionar el botón confirmar, se ejecuta el evento Enter, el cual realiza varias tareas.

```
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
```

En primer lugar se invoca al método NewReservation para crear una reserva en la base de datos. Luego, utilizando **Tipos de datos de workflow**, se inicia el proceso FlightTicketReservation y se marca la tarea TicketReservation como completada.

Estos Tipos de Datos, que comienzan con el prefijo Workflow, son tipos de datos GeneXus y permiten que la aplicación pueda interactuar con el motor de Workflow.

Para poder usar un tipo de datos Workflow e interactuar con la API del motor, se debe definir primero una variable con ese tipo de datos. Luego, con la ayuda que nos provee la información contextual, podemos elegir el método o propiedad, que queremos usar.

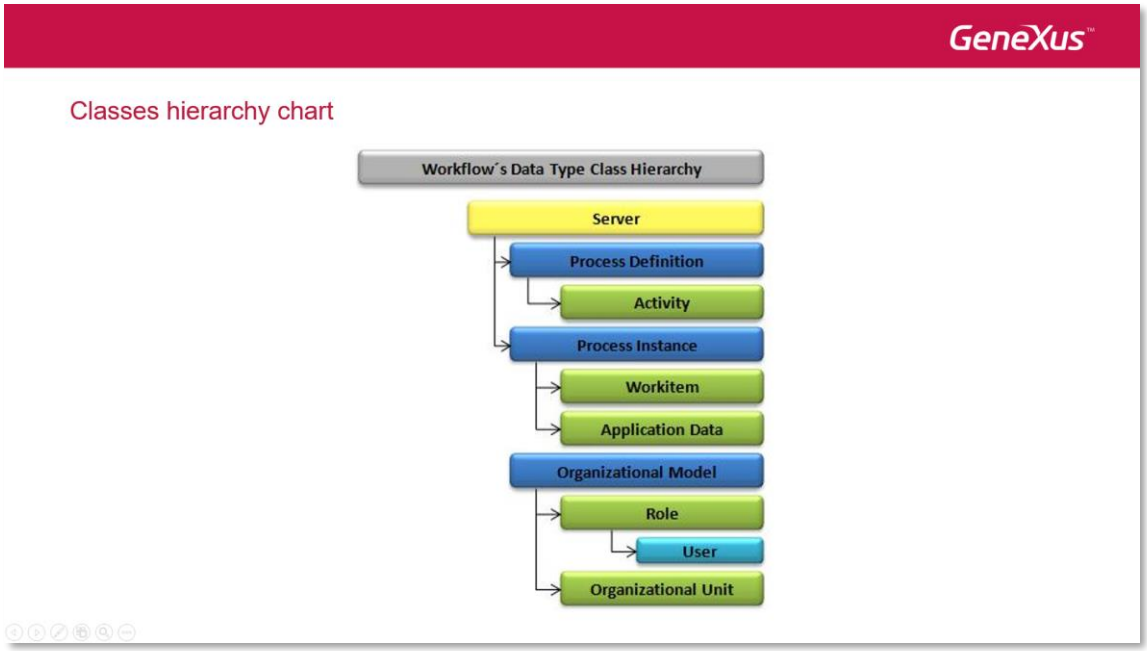
```
Event Enter
    //Create a new reservation with the entered data
    &ReservationId = NewReservation(&ReservationDate,&ReservationQty,&CustomerId,
                     &ReservationDepartureAirportId,&ReservationArrivalAirportId )

    //Create a new FlightTicketReservation process instance
    &WorkflowServer.Connect("WFADMINISTRATOR","WFADMINISTRATOR")
    &WorkflowProcessDefinition = &WorkflowServer.GetProcessDefinitionByName("FlightTicketReservation")
    &WorkflowProcessInstance = &WorkflowProces
    &WorkflowProcessInstance.Subject = 'Flight                    ced 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

Los tipos de datos Workflow están organizados en una jerarquía de clases.

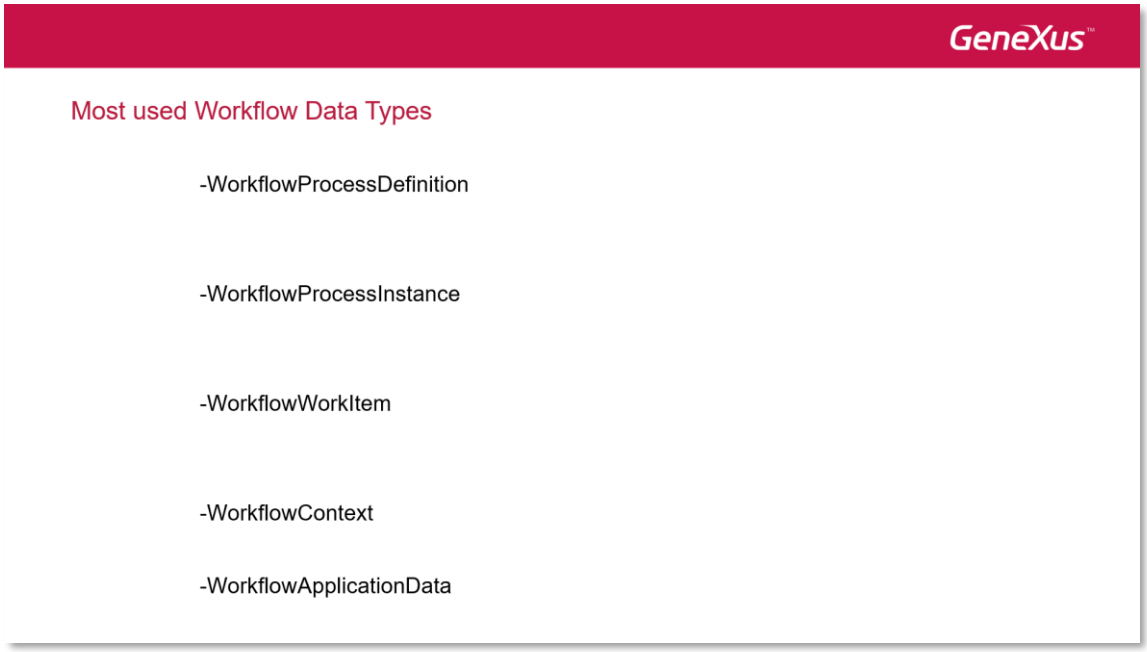


La clase de más jerarquía es la clase **Server**, de la cual dependen 3 clases que son **Process Definition**, la cual nos permite acceder a los componentes de un diagrama de procesos, **Process Instance**, que nos permite acceder a una instancia de un proceso en ejecución y

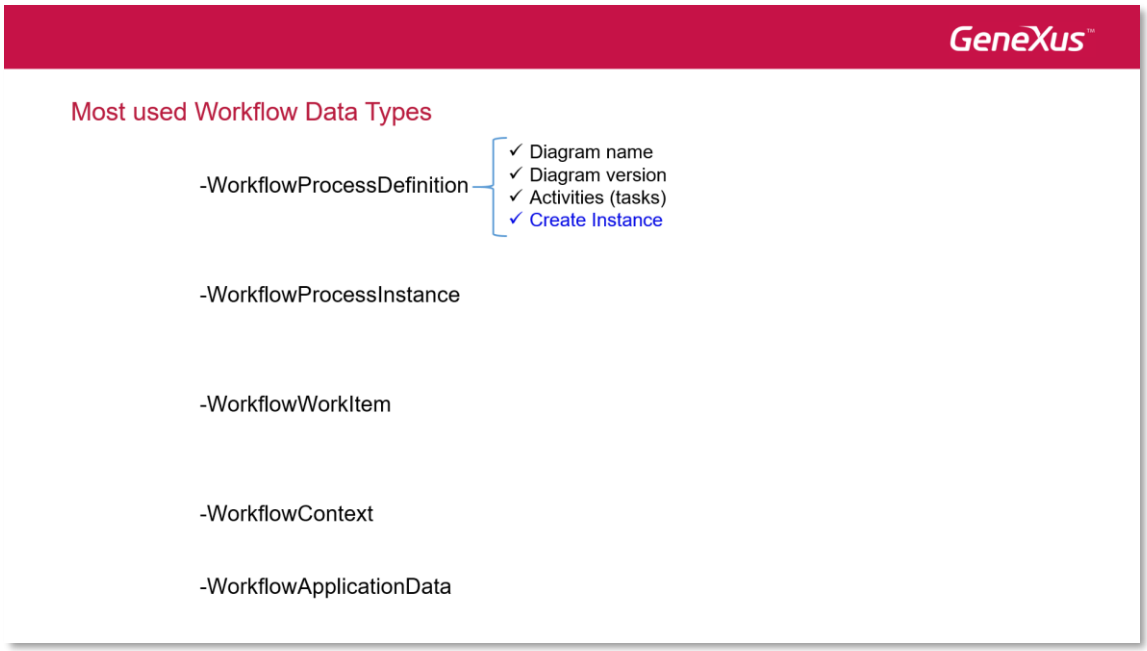
Organization Model, que nos permite acceder a la información de la estructura organizacional de la empresa, como por ejemplo roles y usuarios.

La clase Server es el punto de entrada en la jerarquía de tipos y sus métodos nos permiten acceder a cualquier tipo de datos workflow.

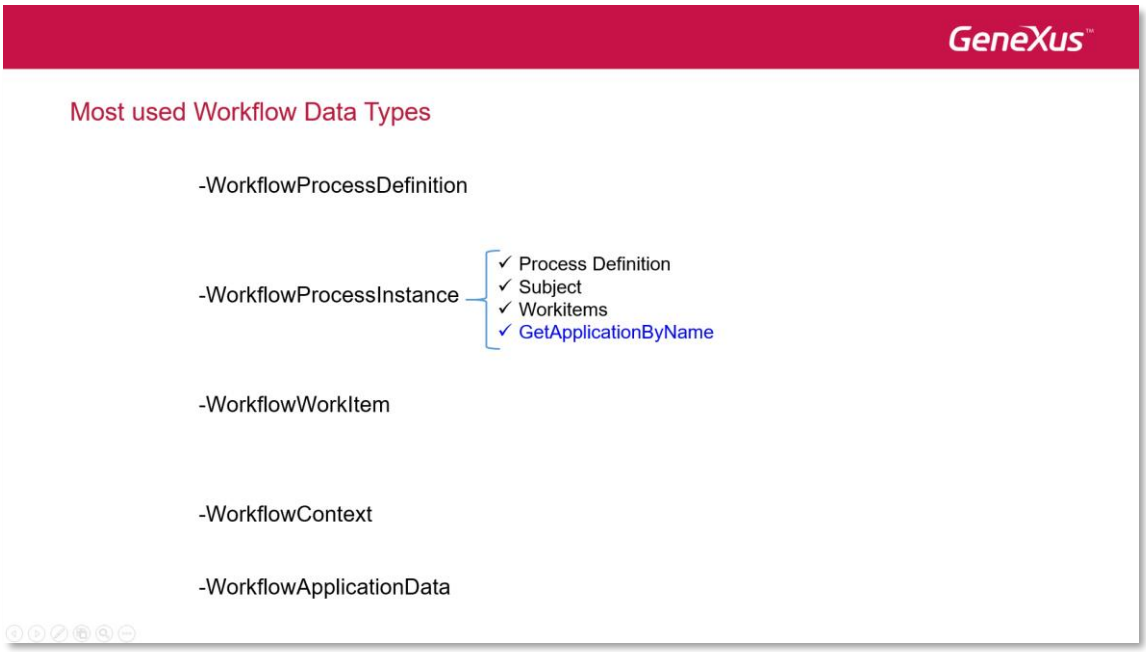
Los tipos de datos más usados son los siguientes:



Mediante **WorkflowProcessDefinition**, podemos acceder a varias propiedades del diagrama, como nombre, versión, tareas que lo incluyen (a las que llamamos actividades) y también es posible mediante el método CreateInstance, crear una instancia en ejecución del proceso, basada en dicha definición.



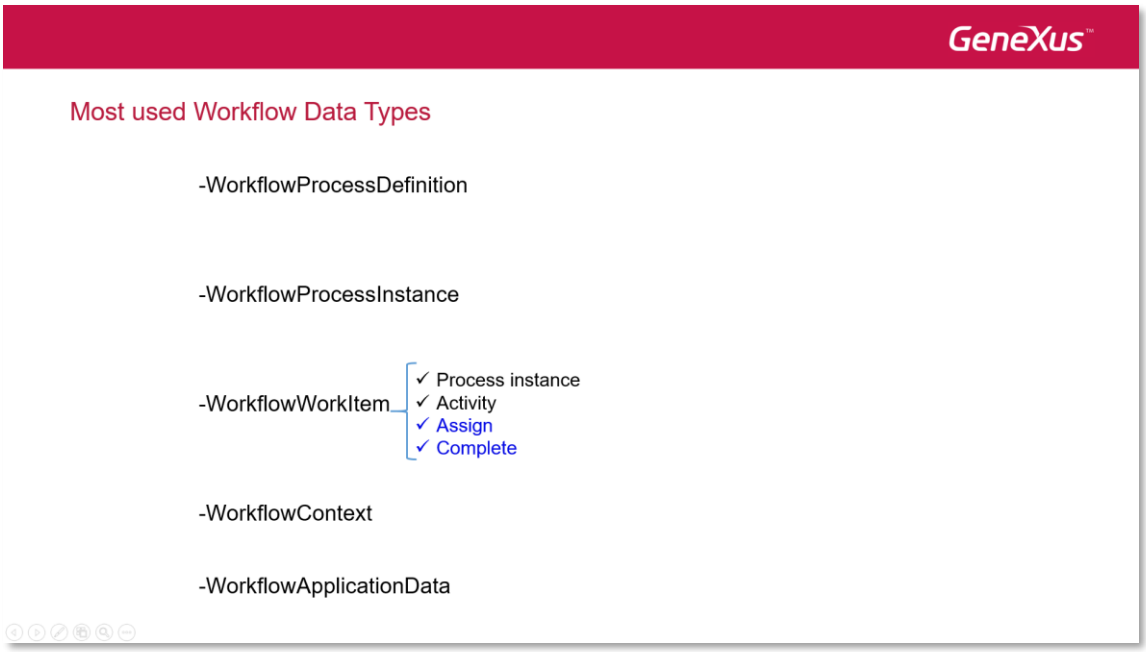
Utilizando WorkflowProcessInstance podemos conocer la definición del proceso en la cual se basa la instancia, el asunto de que se trata la instancia, la colección de los workitems que forman parte de la instancia y a través del método GetApplicationByName podemos recuperar un dato relevante a partir de su nombre.



La clase WorkflowWorkItem nos permite conocer el trabajo a ser realizado por un participante en el contexto de una actividad, dentro de la instancia de proceso.

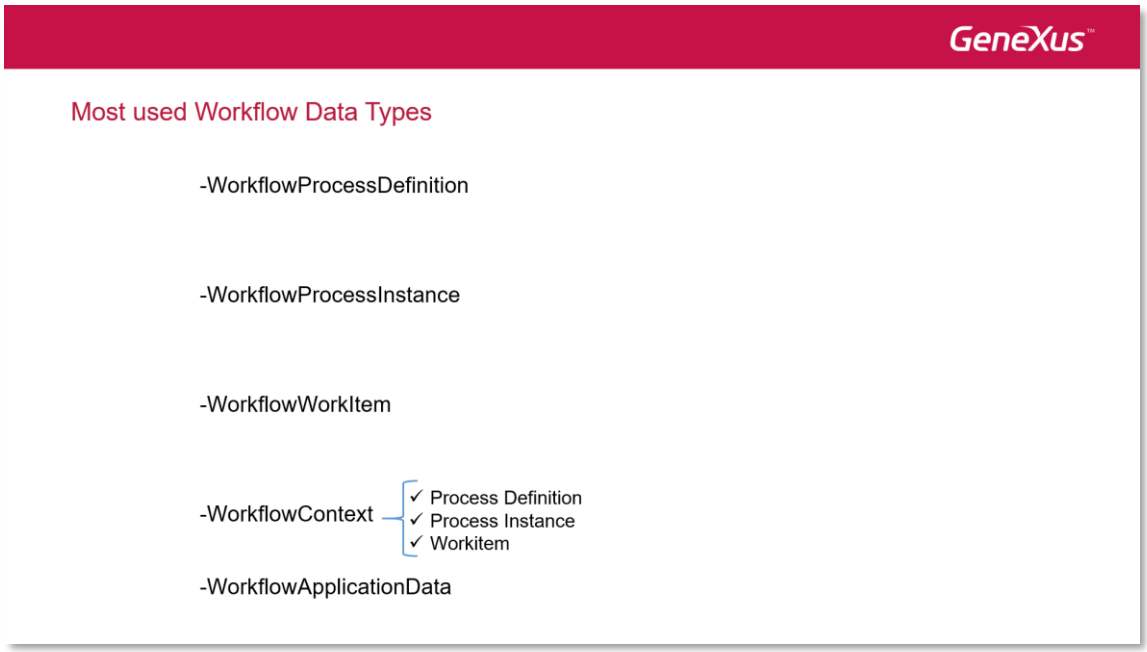
Su propiedad ProcessInstance nos da información de la instancia de proceso al cual pertenece el workitem. La propiedad Activity nos devuelve la actividad que generó el workitem.

El método Assign nos permite asignar un workitem a un usuario determinado y el método Complete permite dar por finalizada la ejecución del workitem.

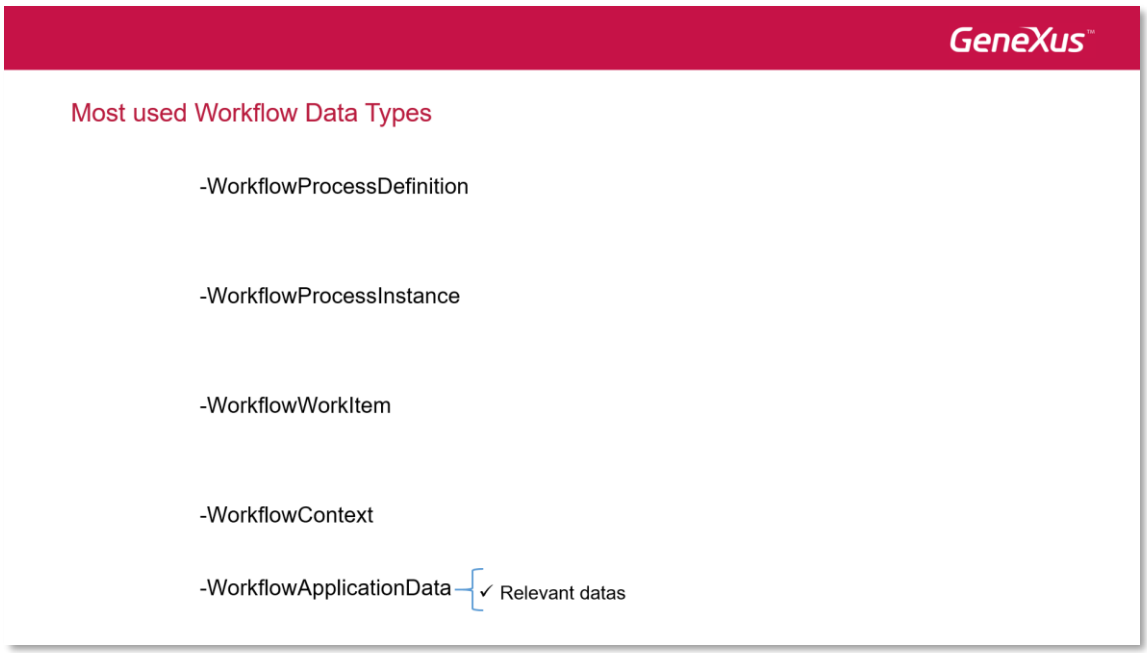


El tipo de datos WorkflowContext nos da información de contexto en la ejecución de una aplicación asociada a una actividad. Este contexto se instancia automáticamente si la aplicación asociada a la actividad (es decir la tarea) es un objeto GeneXus, que forma parte de la misma KB donde se encuentra el diagrama de proceso.

Esta instanciación automática del contexto, nos permite conocer los valores de la definición del proceso, la instancia del proceso y del workitem asociado a la actividad.



Por último el tipo de datos WorkflowApplicationData es el tipo de datos que usamos cuando queremos trabajar con datos relevantes, por ejemplo, cuando almacenamos un dato relevante que obtenemos a través del método GetApplicationDataByName.



Volviendo al evento del webpanel que invoca al proceso FlightTicketReservation, vemos que tenemos definida una variable &WorkflowServer del tipo WorkflowServer. Como práctica,

utilizamos siempre nombres de variables que coincidan con los tipos de datos Workflow, de forma que nos resulte más sencilla su identificación.

La primera operación que realizamos con el tipo de datos WorkflowServer es conectarnos al motor de workflow, utilizando el usuario y la contraseña del administrador.

```
//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
```

Luego obtenemos la definición del proceso FlightTicketReservation a partir de su nombre y lo guardamos en una variable del tipo WorkflowProcessDefinition.

Una vez que tenemos la definición, creamos una instancia de dicho proceso, utilizando el método CreateInstance. Luego cambiamos su asunto, de forma de poder reconocer al proceso fácilmente, en la bandeja de entrada.

A continuación cargamos el dato relevante ReservationId con el identificador de la reserva que creamos previamente y finalmente iniciamos la instancia con el método Start().

Las siguientes líneas de código sirven para marcar la tarea TicketReservation como completada.

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

En primer lugar, a partir de la definición del proceso obtenemos la actividad TicketReservation y con esa actividad obtenemos el workitem correspondiente a la tarea en ejecución en el proceso. Luego marcamos el workitem (es decir la tarea) como completado.

Notemos que luego del método Complete(), hay un Commit. Los cambios realizados utilizando los tipos de datos de workflow están comprendidos dentro de la **Unidad de Trabajo Lógica** de la aplicación.

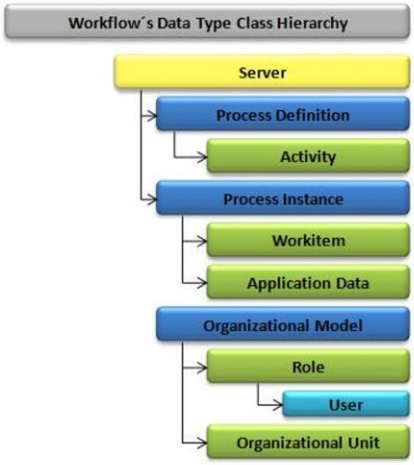
Sin embargo, **las operaciones de workflow no hacen Commit**, por lo que debemos asegurarnos de definir bien la UTL en la aplicación y hacer el Commit al final. En este caso, las operaciones de workflow las hicimos en un webpanel, por lo que es necesario agregar el Commit al finalizar las mismas.

Estos tipos de datos workflow que vimos es un subconjunto de todos los disponibles, por lo que podremos realizar muchas tareas por código, a través de la API del motor de workflow.

Podrá encontrar más información sobre este tema en el siguiente link.

GeneXus

Classes hierarchy chart



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