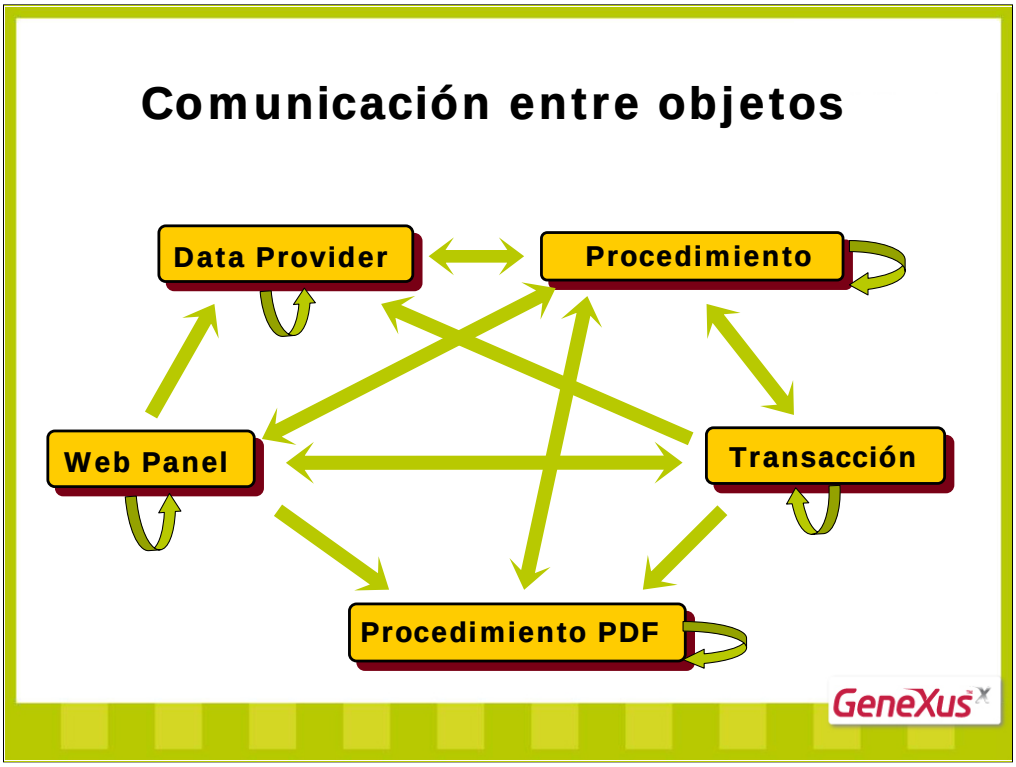
Comunicación entre objetos



Los objetos GeneXus pueden comunicarse entre ellos o con otros programas externos.

Un objeto GeneXus puede llamar o ser llamado por otro objeto, intercambiando información a través de parámetros.

Veremos a continuación cómo invocar desde un objeto a otro, y cómo especificar los parámetros (en el objeto llamador y en el llamado) para el intercambio de la información.

El esquema presentado arriba ilustra las posibles interacciones entre objetos GeneXus para una aplicación Web. Obsérvese que la flecha simple entre Web Panel y Procedimiento PDF (así como entre Transacción y Procedimiento PDF) indica que un Web Panel podrá invocar a un Procedimiento PDF pero un Procedimiento PDF no podrá invocar a un Web Panel (o transacción Web).

CALL - Permite invocar a un objeto GeneXus o a un programa externo, **tanto sin pasarle parámetros, como pasándole**.

UDP (User Defined Procedure) - Permite invocar a un objeto GeneXus o programa externo **tanto sin pasarle parámetros como pasándole**, **y con la particularidad de que el programa llamado retornará necesariamente al menos un valor al programa que lo invocó**. En ambientes Web, un objeto con interfaz una vez invocado no devuelve el control al llamdor, por lo que UDP se utiliza únicamente para invocar a Procedimientos y Data Providers (debido a que estos cumplen la condición de ejecutar y devolver el control al llamador).

Una invocación (ya sea con CALL o UDP) podrá escribirse en distintas partes del objeto llamador, dependiendo de si el mismo es una transacción, web panel, procedimiento, etc.

A su vez UDP puede utilizarse también en la definición de un atributo fórmula. Es decir, se define que cierto atributo es una fórmula y que la definición de la misma consiste en la invocación a un procedimiento utilizando UDP.

Cuando en la sintaxis de la invocación se escribe el nombre del objeto invocado y ningún método de invocación, se asume que se está invocando con udp, por lo que puede omitirse y escribirse directamente:

att|&var = *PgmName*(par_1, ..., par_N)

PARM – Cuando un objeto es invocado desde otro con parámetros, debe tener declarada la lista de parámetros que recibe. Esta declaración se realiza mediante la regla: **PARM**.

A continuación daremos más detalles acerca del uso de CALL, UDP y PARM.

Aquí mostramos un ejemplo de uso de CALL para realizar una invocación y otro ejemplo de uso de UDP.

Dependiendo de qué objeto llamador se trate, estas invocaciones podrán escribirse en una sección u otra del mismo, pero independientemente de eso, aquí apuntamos a mostrar que CALL permite invocar a un objeto con estilo de invocación a un programa, mientras que UDP invoca a un objeto con estilo de invocación a una función.

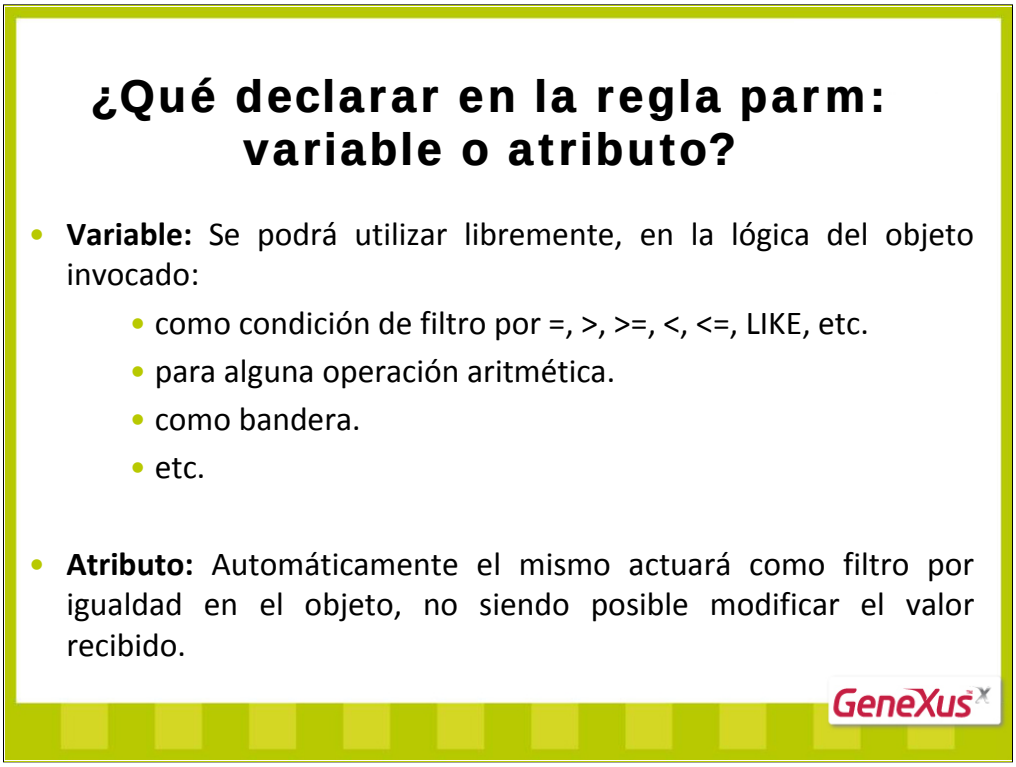
En el primer ejemplo se está utilizando CALL para invocar a un procedimiento pdf (objeto ListInvoice) pasándole un parámetro (*InvoiceId*). En el procedimiento invocado se ha declarado el parámetro que recibe (en su sección de reglas, mediante la regla parm).

En el segundo ejemplo se está utilizando UDP para invocar a un procedimiento (objeto GetDiscount) pasándole dos parámetros (*ProductId, CustomerId*). Ahora, observemos en la sintaxis de la invocación al procedimiento, que el mismo retorna un valor (en la variable *&disc*). Por este motivo, en el procedimiento invocado se han declarado tres parámetros utilizando la regla parm: los dos parámetros recibidos + el parámetro de retorno en último lugar.

Podemos ver entonces que cuando se utiliza CALL para invocar a un objeto enviándole N parámetros, se deben declarar los N parámetros (posicionales y del mismo tipo de datos que los enviados) en el objeto invocado mediante la regla parm.

En cambio cuando se utiliza UDP para invocar a un objeto enviándole N parámetros (a menos que se trate del caso particular de un Data Provider, caso que veremos más adelante):

- en la regla parm del objeto invocado se deben declarar N + 1.
- El último parámetro declarado en la regla parm del objeto invocado corresponde al que se encuentra al principio de todo en la invocación, es decir, al que recibe el valor retornado.
- En algún lugar del objeto invocado se le deberá asignar valor al parámetro de retorno.

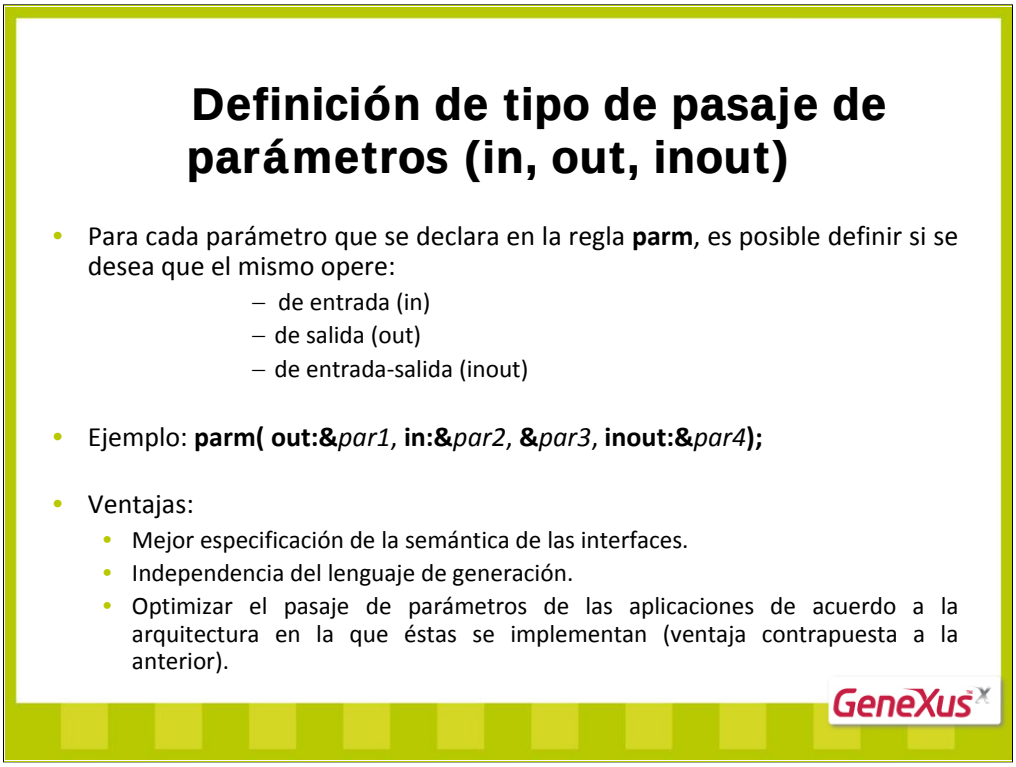


Al definir una invocación a un objeto (ya sea utilizando CALL o UDP), si se tienen que enviar datos por parámetro al objeto invocado, resulta evidente determinar si enviar atributos y/o variables: si un dato a ser enviado por parámetro, se encuentra en el objeto invocador, en un atributo, habrá que enviar el mismo; y si se encuentra en una variable, habrá que enviar la variable.

Sin embargo, al declarar la lista de parámetros en el objeto invocado, el programador GeneXus deberá decidir para cada parámetro, si declararlo mediante un atributo o una variable, independientemente de cómo haya sido enviado.

¿Cuál es la diferencia entre declarar un parámetro como variable o como atributo en la regla **parm** del objeto invocado? Si se declara una variable, se la podrá utilizar libremente en la lógica del objeto invocado: se la podrá utilizar como condición de filtro por igualdad, por mayor, mayor o igual, menor, menor o igual, LIKE, etc.; se la podrá utilizar para alguna operación aritmética, como bandera, o lo que se necesite. Si en cambio se declara un atributo, automáticamente el mismo actuará como filtro por igualdad en el objeto, no siendo posible modificar el valor recibido.

Cuando lleguemos a la etapa del curso en la cual podamos invocar a procedimientos pdf para listados, pasándoles parámetros, así como a otros objetos, podremos terminar de comprender mejor este tema.



Como se puede percibir claramente en la sintaxis del ejemplo, el primer parámetro definido es **de salida**, el segundo parámetro es **de entrada**, y el cuarto parámetro es **de entrada-salida**. Cuando no se especifica nada, como es el caso del tercer parámetro de la sintaxis, dependerá de lo siguiente:

• si el objeto fue invocado con CALL, el parámetro, será de **entrada-salida**.

• si el objeto fue invocado con UDP, y se trata del último parámetro, será de **salida**; y si se trata de otro parámetro distinto del último, dependerá del lenguaje de generación.

Declarar explícitamente cómo se desea que cada parámetro opere, tiene las siguientes ventajas:

1. Mejor especificación de la semántica de las interfaces; es decir, tanto para GeneXus como para el programador cuando trabaje con un objeto, será claro:

- si el mismo vendrá con valor y luego de la ejecución del objeto invocado, se devolverá al objeto invocador el valor con que haya quedado (inout).
- si el mismo vendrá con valor y luego de la ejecución del objeto invocado, no se devolverá al objeto invocador el valor con que haya quedado (in).
- si el mismo no vendrá con valor y luego de la ejecución del objeto invocado, se devolverá al objeto invocador el valor que tenga (out).

2. Independencia del lenguaje de generación; es decir, si se define explícitamente cómo se desea que cada parámetro opere, al generar las aplicaciones utilizando diferentes lenguajes de generación no estará cambiando el comportamiento de los parámetros en base al comportamiento por defecto del lenguaje de generación correspondiente.

3. Optimizar el pasaje de parámetros de acuerdo a la arquitectura en la que éstas se generen (siendo una ventaja contrapuesta a la anterior); esto se refiere a lo siguiente: para la mayoría de los lenguajes es más eficiente pasar los parámetros por referencia (inout) que por valor (in / out); pero en Java, por ejemplo, los parámetros solo se pueden pasar por valor, por lo que para lograr la funcionalidad de pasarlos por referencia es necesario hacer conversiones de parámetros, lo cual puede redundar en un overhead importante; por otro lado, cuando se trata de aplicaciones distribuidas (por ejemplo Java con RMI o HTTP), la utilización de parámetros de tipo out tiene la ventaja de que no es necesario enviar al parámetro en la invocación, a diferencia de si los parámetros se definen de inout (que implica que haya que pasar todos los parámetros); esto tiene como consecuencia que se envíen más bytes de los necesarios, lo cual es inconveniente especialmente en entornos tales como Internet.

La **función Link** se asocia a la propiedad link de un control dentro de cualquier evento de una transacción o web panel, teniendo como resultado que al hacer clic sobre dicho control se realizará la llamada al objeto o URL referenciada en el link.

*PgmName (*el objeto invocado) podrá ser un web panel, transacción, o procedimiento PDF[1].

---

[1] También un procedimiento HTTP, pero no profundizaremos sobre este concepto en este curso

El **comando Link** puede ser utilizado dentro de cualquier evento de una transacción o web panel[1].

Cuando se ejecute el evento, al llegar a la sentencia con el **comando Link**, se redireccionará en forma automática a la URL especificada.

En caso de utilizarse el **comando Link** como en el ejemplo 1, invocando a un *PgmName (*siendo *PgmName* un web panel, transacción o procedimiento PDF), será equivalente a la utilización de Call. Opcionalmente se podrán pasar parámetros al objeto invocado, debiendo declararse los mismos en el objeto llamado, con la regla parm.

---

[1] También un procedimiento HTTP, pero no profundizaremos sobre este concepto en este curso.