


Behavior Client side events grammar



23-Behavior,Client-side-events-grammar_sp



Developing the mobile application

Behavior: client-side events grammar

Cecilia Fernández | GeneXus Training

En este video, abordaremos la gramática de los eventos del cliente, es decir, todo lo que allí se puede escribir.

Primero que nada, resumiremos las invocaciones que pueden realizarse dentro de un evento de un cliente.

Invocations

Commands

Y luego, veremos los otros comandos, recordando lo que decíamos antes: la gramática de los eventos del cliente, es reducida respecto a la de los eventos del server.

No todo lo que escribimos en un evento Start, Refresh y Load, puede ser escrito en un evento del cliente.

Sobre las invocaciones, volveremos en otro video, para estudiar las Call Options

Invocations

CallOptions

Commands

que permiten indicar algunos aspectos de invocación a ser ejecutada inmediatamente.... Como los efectos de entrada y salida de la pantalla invocada...

Invocations

CallOptions EnterEffect & ExitEffect

Commands

el lugar de la pantalla donde se abrirá el objeto invocado...

Invocations

CallOptions EnterEffect & ExitEffect
 Target & TargetSize

Commands

O incluso la relación con el stack

Invocations

CallOptions EnterEffect & ExitEffect
 Target & TargetSize
 Type Push/Replace

Commands

Y hasta podemos especificar si la pantalla llamada funcionará como callout



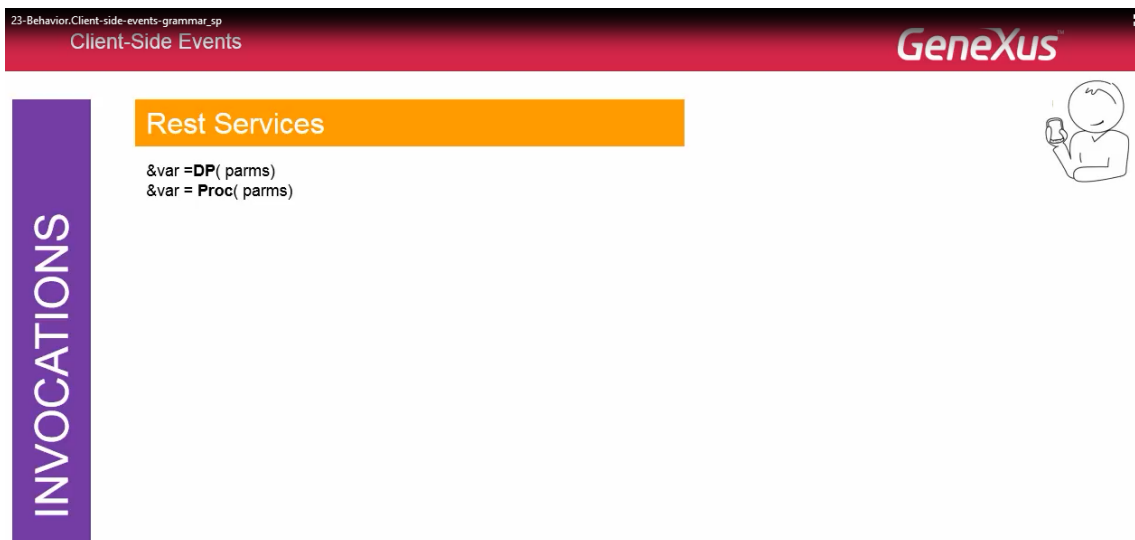
Invocations

CallOptions EnterEffect & ExitEffect
 Target & TargetSize
 Type Push/Replace
 Callout/Popup

Commands

o incluso como Popup.

Podemos clasificar las invocaciones



como servicios Rest del servidor.. como Data Providers o procedimientos.. que nos devuelvan información, que cargaremos en una variable, en el dispositivo.

Necesariamente deben estar expuestos como servicios Rest. No podemos llamar a un procedimiento interno **desde el dispositivo** si estamos en una aplicación de arquitectura online. Al final del curso veremos el caso offline.

También podemos querer dentro de un evento del lado del cliente, ingresar un nuevo registro a la base de datos, sin tener que pedir información al usuario.



INVOCATIONS

Rest Services

```
&var =DP( parms)
&var = Proc( parms)

BC - Batch    &bc.Load (pk)
               &bc.Att1 = ...
               &bc.Att2 = ...
               &bc.Save() //&bc.Delete()
```

Esto se hace como en cualquier otro objeto GeneXus, con los métodos y propiedades del business component; a excepción de que aquí también deberá estar expuesto como servicio Rest, dado que estamos invocando desde el dispositivo, en una arquitectura online.

Dentro de un evento del lado del cliente, también podemos llamar a la pantalla de Detail del Work With, para insertar, actualizar o eliminar



INVOCATIONS

Rest Services

```
&var =DP( parms)
&var = Proc( parms)

BC - Batch    &bc.Load (pk)
               &bc.Att1 = ...
               &bc.Att2 = ...
               &bc.Save() //&bc.Delete()
```

Work With - Edit BC - Interactive

```
<WorkWithObject>.<levelname>.Detail.Insert( &bc )
<WorkWithObject>.<levelname>.Detail.Update( primaryKey )
<WorkWithObject>.<levelname>.Detail.Delete( primaryKey )
```



lo que internamente se traducirá en una invocación al business component: Rest.

Aquí, a través de la pantalla, se le piden los datos al usuario.. y luego se realiza esa operación de manera transparente para el desarrollador.

También podríamos simplemente querer llamar al List o al Detail en modo View

23-Behavior.Client-side-events-grammar.sp Client-Side Events **GeneXus**

INVOCATIONS

Rest Services

```
&var =DP( parms)      BC - Batch  &bc.Load( pk)
&var = Proc( parms)    &bc.Att1 = ...
                        &bc.Attn = ...
                        &bc.Save() //&bc.Delete()
```

Work With - Edit BC - Interactive


```
<WorkWithObject>.<levelname>.Detail.Insert( &bc )
<WorkWithObject>.<levelname>.Detail.Update( primaryKey )
<WorkWithObject>.<levelname>.Detail.Delete( primaryKey )
```

Work With - View

```
<WorkWithObject>.<levelname>.List()
<WorkWithObject>.<levelname>.Detail( PK )
```

Edit

View



así como a objetos “panels for Smart devices”

23-Behavior.Client-side-events-grammar.sp Client-Side Events **GeneXus**

INVOCATIONS

Rest Services

```
&var =DP( parms)      BC - Batch  &bc.Load( pk)
&var = Proc( parms)    &bc.Att1 = ...
                        &bc.Attn = ...
                        &bc.Save() //&bc.Delete()
```

Work With - Edit BC - Interactive

```
<WorkWithObject>.<levelname>.Detail.Insert( &bc )
<WorkWithObject>.<levelname>.Detail.Update( primaryKey )
<WorkWithObject>.<levelname>.Detail.Delete( primaryKey )
```


Work With - View

```
<WorkWithObject>.<levelname>.List()
<WorkWithObject>.<levelname>.Detail( PK )
```

Panels for Smart Devices

Edit

View



que son pantallas un poco más flexibles que las de los work with.

También podemos llamar a un Dashboard

Client-Side Events
GeneXus™

INVOCATIONS

Rest Services			
<pre>&var =DP(parms) &var = Proc(parms)</pre>	<pre>BC - Batch</pre>	<pre>&bc.Load(pk) &bc.Att1 = ... &bc.Attn = ... &bc.Save() //&bc.Delete()</pre>	
Work With - Edit BC - Interactive			
<pre><WorkWithObject>.<levelname>.Detail.Insert(&bc) <WorkWithObject>.<levelname>.Detail.Update(primarykey) <WorkWithObject>.<levelname>.Detail.Delete(primarykey)</pre>			
Work With - View			
<pre><WorkWithObject>.<levelname>.List() <WorkWithObject>.<levelname>.Detail(PK)</pre>			
Panels for Smart Devices			<div style="font-size: 2em; color: #808000;">}</div> <div style="color: #808000; font-weight: bold;">Edit</div> <div style="font-size: 2em; color: #808000;">}</div> <div style="color: #808000; font-weight: bold;">View</div>
Dashboard			

O utilizar alguna de las funcionalidades provistas por las apis

Client-Side Events
GeneXus™

INVOCATIONS

External Objects			
<pre>Msg(&var) Confirm(&var) Return Refresh AddressBook.AddContact(...)</pre>	<pre>//Interop //interop //SDActions //SDActions</pre>		

como desplegar un mensaje en la pantalla... pedir confirmación al usuario para continuar... volver al llamador... refrescar la pantalla... agregar un contacto a la libreta de direcciones... etc.

También se puede invocar a un web panel. Este se abrirá en el navegador del dispositivo, al que se le quita el marco para que luzca más parecido al resto de la aplicación.

Eso en cuanto a las invocaciones. Los comandos aceptados por el momento, son los que se muestran. Por ejemplo, puede hacerse visible o invisible un control... se le puede configurar la clase...

INVOCATIONS

External Objects

```
Msg( &var)           //Interop
Confirm( &var )       //Interop
Return               //SDActions
Refresh              //SDActions
AddressBook.AddContact(...)
```



Web Panels

COMMANDS

```
Composite
<Control> <Property> = <value>
If <Bool_expr>
Do while <Bool_expr>
Do-sub (except Dashboard)
For each selected line
Simple variable assignation
    &var = <expr>
SDT or BC elements assignation
    &SDT.A = <value>
    &BC.A = <value>
```

Se pueden utilizar las estructuras de control: if y do while...

INVOCATIONS

External Objects

```
Msg( &var)           //Interop
Confirm( &var )       //Interop
Return               //SDActions
Refresh              //SDActions
AddressBook.AddContact(...)
```



Web Panels

COMMANDS

```
Composite
<Control> <Property> = <value>
If <Bool_expr>
Do while <Bool_expr>
Do-sub (except Dashboard)
For each selected line
Simple variable assignation
    &var = <expr>
SDT or BC elements assignation
    &SDT.A = <value>
    &BC.A = <value>
```

Por ahora no están incluidas, las:

- For in
- Do case
- Y los métodos como el Add de los SDTs o Business Components

INVOCATIONS

External Objects

Msg(&var) //Interop
Confirm(&var) //interop
Return //SDActions
Refresh //SDActions
AddressBook.AddContact(...)



Web Panels

COMMANDS

Composite
<Control>.<Property> = <value>
If <Bool_expr>
Do while <Bool_expr>
Do-sub (except Dashboard)
For each selected line
Simple variable assignment
&var = <expr>
SDT or BC elements assignment
&SDT.A = <value>

For ... in...
Do case...

INVOCATIONS

External Objects

Msg(&var) //Interop
Confirm(&var) //interop
Return //SDActions
Refresh //SDActions
AddressBook.AddContact(...)



Web Panels

COMMANDS

Composite
<Control>.<Property> = <value>
If <Bool_expr>
Do while <Bool_expr>
Do-sub (except Dashboard)
For each selected line
Simple variable assignment
&var = <expr>
SDT or BC elements assignment
&SDT.A = <value>
&BC.A = <value>

&STD.Add(...)

INVOCATIONS

External Objects

Msg(&var) //Interop
Confirm(&var) //interop
Return //SDActions
Refresh //SDActions
AddressBook.AddContact(...)



Web Panels

COMMANDS

Composite
<Control>.<Property> = <value>
If <Bool_expr>
Do while <Bool_expr>
Do-sub (except Dashboard)
For each selected line
Simple variable assignment
&var = <expr>
SDT or BC elements assignment
&SDT.A = <value>
&BC.A = <value>

&BC.Add(...)

Estas estructuras, ahora, aceptan expresiones booleanas, que pueden incluir todo lo conocido

INVOCATIONS

External Objects

```
Msg( &var)           //Interop
Confirm( &var )       //Interop
Return               //SDActions
Refresh              //SDActions
AddressBook.AddContact(...)
```



Web Panels

COMMANDS

```
Composite
<Control>.<Property> = <value>
If <Bool_expr>
Do while <Bool_expr>
Do-sub (except Dashboard)
For each selected line
Simple variable assignation
    &var = <expr>
SDT or BC elements assignation
    &SDT.A = <value>
    &BC.A = <value>
```

Inside an expression:

```
Variables
Attributes
Constants
Methods
Functions
Control properties
Operators (+, -, *, /, ^)
```

Not allowed: udp or external objects

excepto invocaciones udp's o métodos de external objects.

También pueden utilizarse invocaciones a subrutinas excepto en objeto Dashboard.

En el comando For each selected line

INVOCATIONS

External Objects

```
Msg( &var)           //Interop
Confirm( &var )       //Interop
Return               //SDActions
Refresh              //SDActions
AddressBook.AddContact(...)
```



Web Panels

COMMANDS

```
Composite
<Control>.<Property> = <value>
If <Bool_expr>
Do while <Bool_expr>
Do-sub (except Dashboard)
For each selected line
Simple variable assignation
    &var = <expr>
SDT or BC elements assignation
    &SDT.A = <value>
    &BC.A = <value>
```

Inside an expression:

```
Variables
Attributes
Constants
Methods
Functions
Control properties
Operators (+, -, *, /, ^)
```

Not allowed: udp or external objects

sólo se puede invocar a un proc. Si es online se invoca una vez y es en el server en el que se ejecuta N veces (por cada línea seleccionada) y se devuelve el resultado final en un json.

En las asignaciones a variable simple, se le puede asignar una expresión.

INVOCACTIONS

External Objects

Msg(&var) //Interop
Confirm(&var) //Interop
Return //SDActions
Refresh //SDActions
AddressBook.AddContact(...)



Web Panels

COMMANDS

Composite
<Control>.<Property> = <value>
If <Bool_expr>
Do while <Bool_expr>
Do-sub (except Dashboard)
For each selected line
Simple variable assignment
&var = <expr>
SDT or BC elements assignment
&SDT.A = <value>
&BC.A = <value>

Inside an expression:
Variables
Attributes
Constants
Methods
Functions
Control properties
Operators (+, -, *, /, ^)

Not allowed: udp external objects

o una invocación a un proc, que devuelva el valor... pero a un elemento de SDT o Business Component, sólo se le puede asignar 1 valor.. no una expresión.

Respecto al comando Composit, recordemos lo que ya habíamos adelantado en otros videos. Cuando deben realizarse un par de invocaciones o más en un evento, es obligatorio agrupa el código completo del evento, dentro de este comando.

De este modo, cuando ocurra un error en la secuencia de llamadas

INVOCACTIONS

External Objects

Msg(&var) //Interop
Confirm(&var) //Interop
Return //SDActions
Refresh //SDActions
AddressBook.AddContact(...)



Web Panels

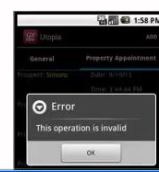
COMMANDS

Composite
<Control>.<Property> = <value>
If <Bool_expr>
Do while <Bool_expr>
Do-sub (except Dashboard)
For each selected line
Simple variable assignment
&var = <expr>
SDT or BC elements assignment
&SDT.A = <value>
&BC.A = <value>

Inside an expression:
Variables
Attributes
Constants
Methods
Functions
Control properties
Operators (+, -, *, /, ^)

Not allowed: udp external objects

Composite Automatic Error Handling



la ejecución se detiene y se manejan los errores automáticamente desplegándolos en la pantalla sin tener que implementar ninguna programación. **Este comando está implementado sólo en Smart Devices y es obligatorio en estos.**

Lo estudiaremos en detalle en el video que sigue.

Acompáñenos.

