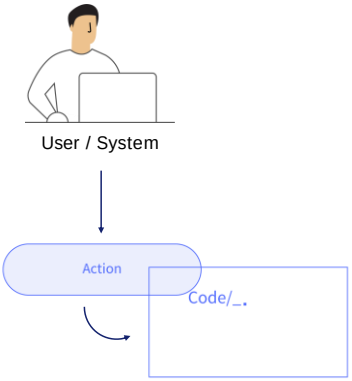
Más sobre Eventos en Transacciones

GeneXus

Evento: Concepto



Anteriormente, hemos estudiado que el objeto Transacción dispone de eventos que podemos programar para controlar y definir su comportamiento.

A modo de repaso, recordemos los eventos disponibles:

Eventos en Transacciones: Start event


```
Event Start
     /* Generated by Work With Pattern [Start] - Do not change */
     [web]
     {
     If not IsAuthorized(&PgmName)
          NotAuthorized(&PgmName)
     Endif

     &TrnContext.FromXml(&WebSession.Get(!"TrnContext"))
     &Insert_CountryId.SetEmpty()
     &Insert_CityId.SetEmpty()
     &Insert_CategoryId.SetEmpty()

     }
     /* Generated by Work With Pattern [End] - Do not change */
Endevent
```

El evento **Start**, que permite definir una acción a ser ejecutada cuando se abre y comienza a trabajarse con la transacción

Eventos en Transacciones: TrackContext event


```
Event TrackContext(&AttractionId)
     WCDetails.Object = AttractionsDetail.Create(&AttractionId)
Endevent
```

Si cambia el valor de la variable &AttractionId, se disparará este evento, lo que provocará que el componente en pantalla WCDetails se vuelva a cargar a partir del Web Component AttractionDetail, con el nuevo valor de la variable.

El evento **Track Context**, que permite programar qué acción realizar cuando hay algún cambio en el contexto de la ejecución de la transacción.

¿Qué se entiende por Contexto?

Nos referimos al contexto en relación a una pantalla, a un form. Cuando nos movemos dentro de una pantalla, estamos cambiando el contexto de atributos y variables. Esta información del cambio en el contexto es fundamental para definir una acción a tomar.

Eventos en Transacciones: OnMessage event


```
Event OnMessage(&NotificationInfo)
  for each line
    if (&NotificationInfo.Id=&postid.ToString())
     //processs the notification data
    endif
  endfor
EndEvent
```

El evento **OnMessage**, relacionado con las notificaciones web.

Eventos en Transacciones: Insert, Update, Delete events


```
Event Insert(&Messages)
     If DocumentType = Type.Invoice
          &Invoice = New()
          &Invoice.InvoiceId = DocumentId
          &Invoice.InvoiceDate = DocumentDate
          &Invoice.InvoiceTotal = DocumentAmount
          &Invoice.Insert()
          &Messages = &Invoice.GetMessages()
     else
          &Receipt = New()
          &Receipt.ReceiptId = DocumentId
          &Receipt.ReceiptDate = DocumentDate
          &Receipt.ReceiptTotal = DocumentAmount
          &Receipt.Insert()
          &Messages = &Receipt.GetMessages()
     endif
Endevent
```

Los eventos **Insert, Update y Delete**, que, como ya hemos visto, aplican al caso específico de las transacciones dinámicas, y permiten programar su comportamiento para que sean actualizables.

Eventos en Transacciones: Exit event


```
Event Exit
     //Process code
Endevent
```

El evento **Exit**, permite programar las acciones a realizar cuando la transacción ya se cierra.

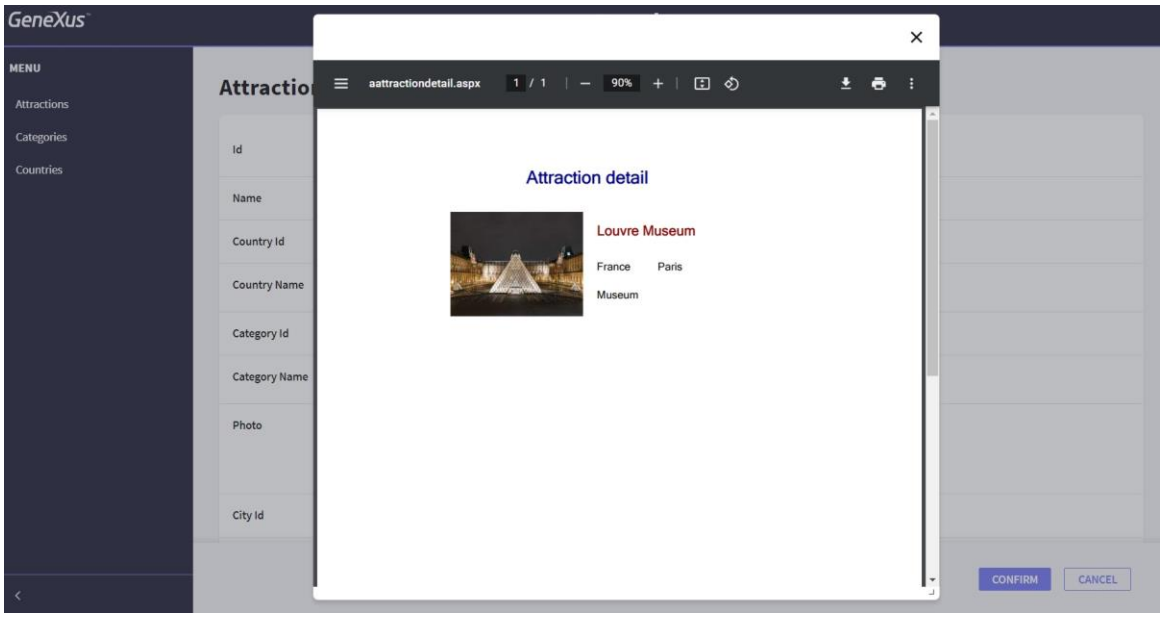
Eventos en Transacciones: After Trn event


```
Event After Trn
      /* Generated by Work With Pattern [Start] - Do not change */
      [web]
      {
      If (&Mode = TrnMode.Delete and not &TrnContext.CallerOnDelete)
           WWAttraction()
      Endif

      Return
      }
      /* Generated by Work With Pattern [End] - Do not change */
EndEvent
```

Y el evento **AfterTrn**, que se dispara luego de ejecutado el Commit. Si recordamos lo ya estudiado sobre los momentos de disparo de reglas, este evento AfterTrn se ejecuta igual al momento de disparo on AfterComplete.

Nuevo requerimiento...



Vamos a resolver ahora un nuevo requerimiento para la Agencia de Viajes.

Nos solicitan que cada vez que se trabaje con el registro de una atracción turística, y luego de efectuado el commit, se muestre una ventana popup con un pdf con el detalle de dicha atracción,

Nuevo requerimiento...

```
Event After Trn
     AttractionDetail.Popup(AttractionId)

     /* Generated by Work With Pattern [Start] - Do not change */
     [web]
     {
     If (&Mode = TrnMode.Delete and not &TrnContext.CallerOnDelete)
          WWAttraction()
     Endif

     Return
     }
     /* Generated by Work With Pattern [End] - Do not change */
EndEvent
```

Ya tenemos definido el listado pdf de nombre AttractionDetail, que recibe por parámetro el identificador de una atracción, y muestra su detalle.
La pregunta entonces a responder ahora es la siguiente: ¿Dónde declaramos la llamada a este procedimiento para lograr la funcionalidad requerida?

Si necesitamos llamarlo luego de efectuado el commit, entonces podríamos pensar por ejemplo en declarar una regla en la transacción Attraction, y condicionarla al momento on AfterComplete. Pero el requerimiento nos pide ver el listado pdf en una ventana popup y este método popup no lo tenemos disponible en las reglas de una transacción.

Por lo tanto, podemos recurrir al evento AfterTrn.

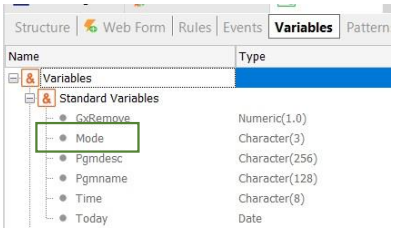
Ahora bien, como tenemos aplicado el pattern Work With, ya sabemos que GeneXus agrega código automáticamente en esta transacción Attraction, y que en particular agrega en este evento AfterTrn el comando de retorno.

Por lo tanto, debemos llamar al listado antes de que se ejecute este comando Return, y por fuera de las marcas del código automáticamente generado por la aplicación del pattern.

Nuevo requerimiento...

```
Event After Trn
    If &Mode = TrnMode.Insert
        AttractionDetail.Popup(AttractionId)
    endif
    /* Generated by Work With Pattern [Start] - Do not change */
    [web]
    {
    If (&Mode = TrnMode.Delete and not &TrnContext.CallerOnDelete)
        WWAttraction()
    Endif

    Return
    }
    /* Generated by Work With Pattern [End] - Do not change */
EndEvent
```



Vale mencionar que podemos condicionar la llamada a este pdf dependiendo del modo de ejecución de la transacción.

Si queremos que solamente se muestre el detalle cuando se está insertando una nueva atracción, y no cuando se modifica o se elimina, entonces podemos poner algo así...

Recordemos que la variable &Mode es una variable del sistema que guarda el modo en que la transacción está siendo ejecutada. El valor de esa variable se recibe en la regla Parm automáticamente declarada por la aplicación del pattern Work With, y depende de la acción seleccionada por el usuario en la pantalla principal.

Eventos en Transacciones - Restricciones

- No es posible actualizar directamente la base de datos

  No es posible asignar valor a un atributo, pero sí es posible llamar a un procedimiento para realizar la actualización necesaria.

- Comando Commit

```
Event Insert
    If DocumentType = Type.Invoice
        &Invoice = New()
        &Invoice.InvoiceId = DocumentId
        &Invoice.InvoiceDate = DocumentDate
        &Invoice.InvoiceTotal = DocumentAmount
        &Invoice.Insert()
    else
        &Receipt = New()
        &Receipt.ReceiptId = DocumentId
        &Receipt.ReceiptDate = DocumentDate
        &Receipt.ReceiptTotal = DocumentAmount
        &Receipt.Insert()
    endif
Endevent
```

Para finalizar, vale mencionar que en los eventos de las transacciones no es posible realizar directamente actualizaciones a la base de datos, si bien es posible trabajar con business components.

Cuando trabajamos con business components, como en el caso de los eventos asociados a las transacciones dinámicas, no declaramos el comando commit ya que estamos en el contexto de una transacción y por lo tanto tenemos la propiedad Commit on exit con el valor True.

GeneXus™

training.genexus.com
wiki.genexus.com