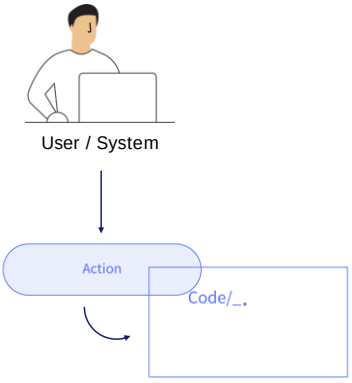
Mais sobre Eventos em Transações

GeneXus

Evento: Conceito



Anteriormente, estudamos que o objeto Transação dispõe de eventos que podemos programar para controlar e definir seu comportamento .

A título de revisão, vamos lembrar os eventos disponíveis:

Eventos em transações: 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
```

O evento **Start**, que permite definir uma ação a ser executada quando é aberta e começa a se trabalhar com a transação.

Eventos em transações: TrackContext event


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

Se você alterar o valor da variável &AttractionId, este evento será disparado, o que fará com que o componente WCDetails na tela seja recarregado do Web Component AttractionDetail, com o novo valor da variável.

O evento **Track Context**, que permite programar qual ação realizar quando houver alguma mudança no contexto da execução da transação.

O que se entende por Contexto?

Nos referimos ao contexto em relação a uma tela, a um form. Quando nos movemos dentro de uma tela, estamos mudando o contexto de atributos e variáveis. Esta informação da mudança no contexto é fundamental para definir uma ação a tomar.

Eventos em transações: OnMessage event


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

O evento **OnMessage**, relacionado com as notificações web.

Eventos em transações: 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
```

Os eventos **Insert, Update e Delete**, que, como já vimos, se aplicam ao caso específico das transações dinâmicas, e permitem programar seu comportamento para que sejam atualizáveis.

Eventos em transações: Exit event


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

O evento **Exit**, permite programar as ações a serem realizadas quando a transação já se fecha.

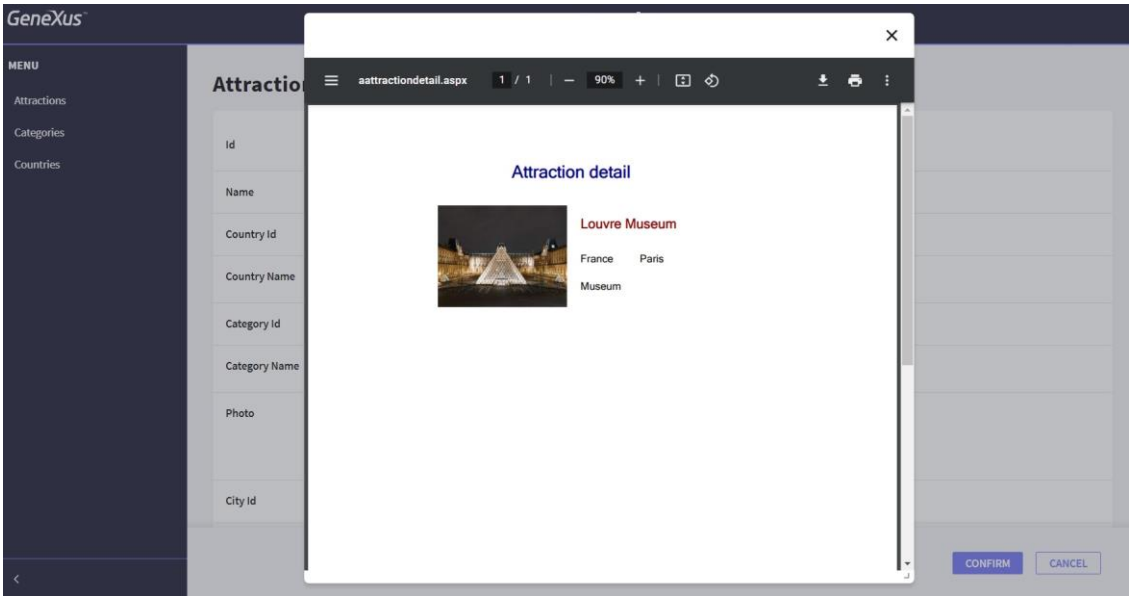
Eventos em transações: After Trn event


```
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
```

E o evento **AfterTrn**, que é disparado depois de executado o Commit. Se nos lembrarmos do que já foi estudado sobre os momentos de disparo de regras, este evento AfterTrn é executado igual ao momento de disparo on AfterComplete.

Novo requisito...



Vamos resolver agora um novo requisito para a Agência de Viagens .

Solicitam-nos que toda vez que se trabalhe com o registro de uma atração turística, e depois de efetuado o commit, seja apresentada uma janela pop-up com um pdf com o detalhe da referida atração,

Novo requisito ...

```
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
```

Já definimos a lista pdf chamada AttractionDetail, que recebe por parâmetro o identificador de uma atração, e mostra seu detalhe.

A pergunta então a responder agora é a seguinte: Onde declaramos a chamada a este procedimento para obter a funcionalidade requerida?
Se precisamos chamá-lo depois de efetuado o commit, então poderíamos pensar, por exemplo, em declarar uma regra na transação Attraction e condicioná-la ao momento on AfterComplete.

Mas o requisito nos pede para ver a lista pdf em uma janela pop-up e este método popup não temos disponível nas regras de uma transação.

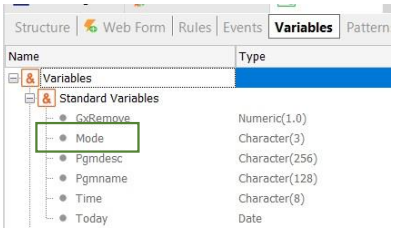
Portanto, podemos recorrer ao evento AfterTrn.
Entretanto, como aplicamos o pattern Work With, já sabemos que GeneXus adiciona código automaticamente nesta transação Attraction e que em particular adiciona neste evento AfterTrn o comando de retorno.

Portanto, devemos chamar a lista antes que seja executado este comando Return, e fora das marcas do código automaticamente gerado pela aplicação do pattern.

Novo requisito ...

```
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 a chamada a este pdf dependendo do modo de execução da transação.

Se quisermos que o detalhe seja mostrado apenas quando está sendo inserida uma nova atração, e não quando é modificada ou eliminada, então podemos colocar algo assim ...

Recordemos que a variável &Mode é uma variável do sistema que armazena o modo no qual a transação está sendo executada. O valor dessa variável é recebido na regra Parm automaticamente declarada pela aplicação do pattern Work With, e depende da ação selecionada pelo usuário na tela principal.

Eventos em transações - Restrições

- Não é possível atualizar o banco de dados diretamente.

  Não é possível atribuir um valor a um atributo, mas é possível chamar um procedimento para realizar a atualização necessária.

- 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 nos eventos das transações não é possível realizar diretamente atualizações na base de dados, embora seja possível trabalhar com business components .

Quando trabalhamos com business components, como no caso dos eventos associados às transações dinâmicas, não declaramos o comando commit, pois estamos no contexto de uma transação e, portanto, temos a propriedade Commit on exit com o valor True
.

GeneXus™

training.genexus.com
wiki.genexus.com