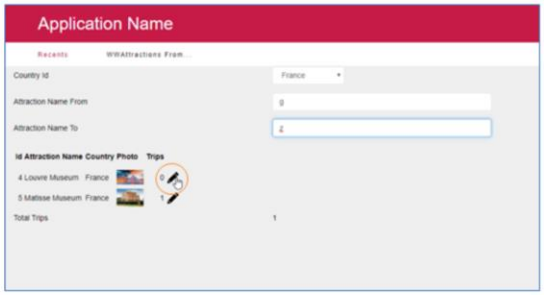
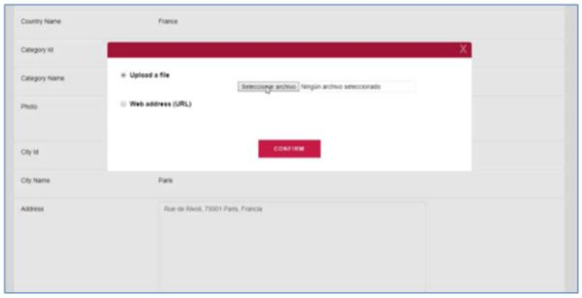
Telas interativas

Como salvar informações de contexto

GeneXus™

Web panel WWAttractionFromScratch






```
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
```

Neste exemplo, veremos uma maneira de manter dados em memória, para evitar que sejam perdidos após chamar outro objeto e, posteriormente retornar a ele.
Para exemplificar esta situação, continuaremos com o exemplo que estamos trabalhando até o momento.
Façamos um pequeno resumo.
No Web Panel que vemos na tela, ao inserir valores nos filtros, o grid é atualizado e mostra apenas os dados que nos interessam, condicionado por esses valores.

Se selecionamos a ação de atualização em uma das linhas, como vimos, nos levará à transação Attraction em modo Update. Onde nos permitirá editar os valores correspondentes à atração selecionada.
Escolheremos uma das atrações para modificar, o faremos por exemplo, com o museu
do Louvre.

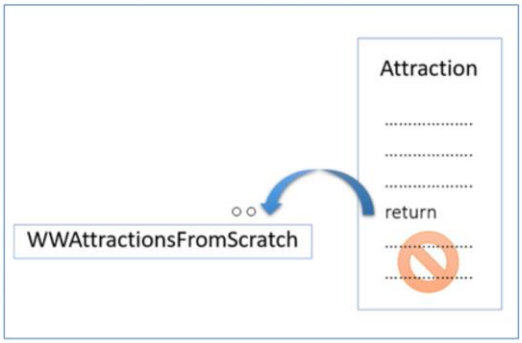
Alteraremos a imagem correspondente ao atributo AttractionPhoto e selecionamos confirmar.

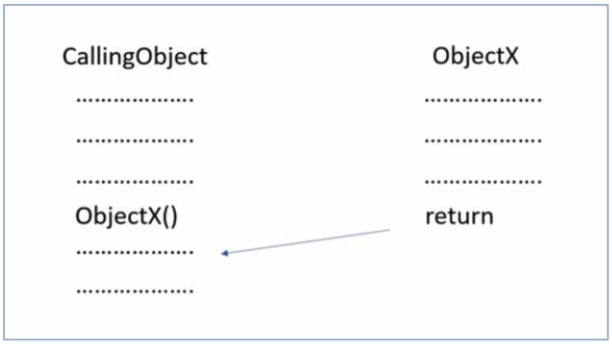
Vemos que esta ação nos retorna ao nosso Web Panel.

Isto ocorre porque o pattern work with aplicado à transação Attraction, automaticamente adicionou o comando Return, para desta forma retornar ao objeto chamador. Isto podemos ver na seção Eventos da transação Attraction, programado dentro do evento “After Trn”.

O evento “After Trn” é disparado quando a transação concluir um ciclo, imediatamente após o commit, ou seja, após ter sido registrado um cabeçalho com suas linhas correspondentes.

Web panel WWAttractionFromScratch






```
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
```

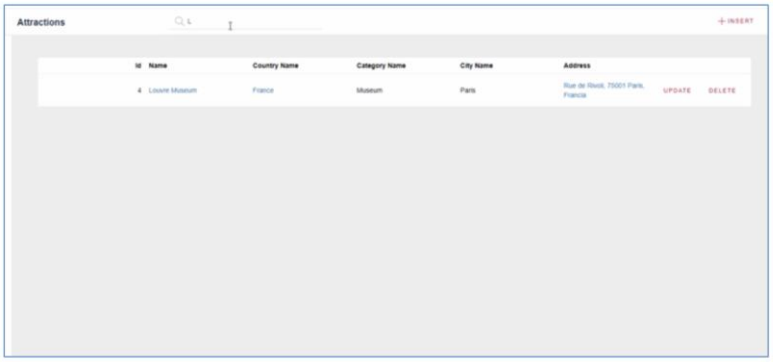
≡ WWAttractionsFromScratch()

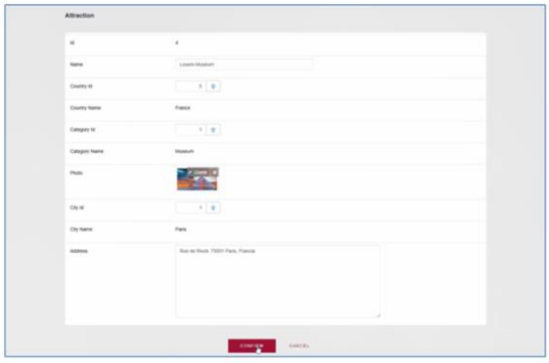
A função do comando return é finalizar a execução que está sendo realizada neste objeto e voltar, retornar ao objeto chamador.
Em nosso exemplo, tanto o Web Panel WWAttractionsFromScratch quanto a transação Attraction, ou seja, tanto o objeto chamador quanto o chamado, são objetos com interface gráfica. Portanto, neste caso, o comando Return é equivalente a se fizéssemos uma invocação ao Web Panel pela primeira vez.
Diferente seria o caso se algum destes dois objetos não tivesse interface gráfica, como é por exemplo o caso de um procedimento. Lá, o comando return do objeto que foi chamado nos retornará para a próxima linha imediata à invocação.

Voltando ao nosso caso, ao retornar serão executados os eventos associados à carga do Web Panel. Primeiro, o evento Start, seguido do Refresh e depois o Load, este último tantas vezes quanto registros houver no grid que atendam às condições declaradas na propriedade conditions. Neste caso, como as condições não se aplicam se as variáveis estiverem vazias, vemos em execução que voltam a ser carregadas todas as atrações.

Por qual motivo isso acontece? Explicaremos em um momento.

Pattern Work With Attraction





Antes veremos o comportamento do Web Panel criado automaticamente a partir da seção patterns da transação Attraction.
Vamos filtrar por nome da atração, digitando a letra “L”, nos mostrará todas as atrações que começam com esta letra, neste caso, a única que existe inserida com esta característica é a do Museu do Louvre.
Selecionemos a ação para atualizar esta atração.

Observamos que nos levará diretamente à transação Attraction e nos mostrará a atração selecionada, permitindo sua edição, exatamente da mesma forma que no caso do nosso web panel implementado manualmente.
Mudaremos a fotografia, confirmamos a atualização e nos retornará ao Web Panel chamador.
Observamos que os filtros são mantidos. Esta é justamente a funcionalidade que queremos alcançar.

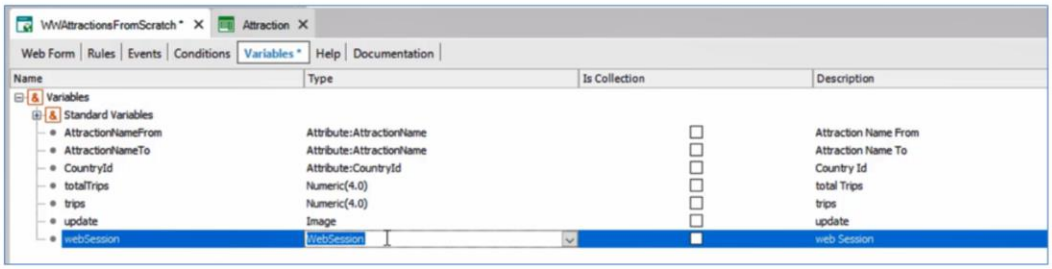
Pattern Work With Attraction

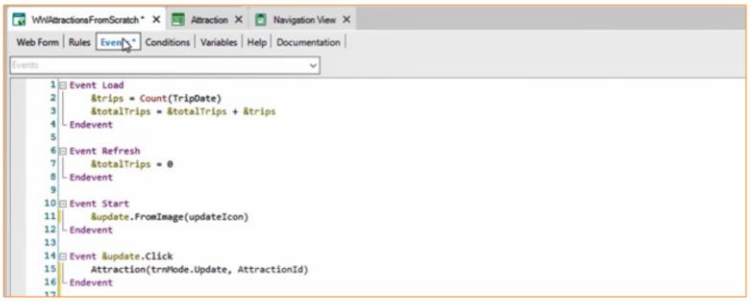
Programaremos em nosso Web Panel uma solução à nossa maneira e, em seguida, revisaremos como o Pattern o faz.
Lembremos que as variáveis declaradas em cada objeto só poderão ser utilizadas dentro dele e enquanto este estiver ativo.

Por exemplo, no nosso caso, quando invocamos a transação Attraction a partir do Web Panel, nesse momento, nosso objeto WWAttractionsFromScratch é destruído e com ele suas variáveis.
Portanto, serão perdidos os valores que tinham guardados. Por outro lado, o que passará a ter um estado ativo é o objeto Attraction.
Então, ao retornar da transação para o Web Panel com o comando return, este último objeto e suas variáveis serão recriados.
É por isso que já não vemos os valores de nossos filtros, porque, na realidade, esse é um novo objeto. E as variáveis que usamos como filtros também foram criadas novamente, as que tínhamos antes de invocar o objeto Attraction foram destruídas.

Isto responde à pergunta que fizemos a um momento atrás, de qual era o motivo pelo qual os valores inseridos em nossos filtros não foram mantidos após a atualização de um registro.
O que precisamos é salvar as informações de cada um de nossos filtros em um tipo de variável global, de modo que não se perca entre execuções, ou seja, na passagem de um objeto a outro. Neste caso, entre o Web Panel e a transação, e da transação novamente ao nosso Web Panel.

WebSession Data Type





Como fazemos então, para poder manter em memória o valor de uma variável?
Temos no GeneXus uma maneira de programar esta funcionalidade. Isso é feito por meio de variáveis do tipo Web Session.
Estas variáveis nos permitem manipular um tipo de conjunto de variáveis globais, nas quais podemos armazenar dados e acessá-los a partir de qualquer objeto, enquanto a sessão esteja ativa.
Isto é exatamente o que estávamos procurando.
Uma grande vantagem destas variáveis é que elas nos permitem salvar um conjunto de dados do tipo chave-valor. Portanto, apenas precisaremos declarar uma variável do tipo Web Session e nela salvar todos os filtros que tenhamos, cada um deles com uma chave única.

Assim, temos uma chave e um valor para o filtro CountryId, outra chave e valor para AttractionNameFrom e, finalmente, para AttractionNameTo, os três filtros que precisamos manter entre execuções do nosso painel

Isto atende à nossa necessidade, poder salvar temporariamente as informações inseridas nos filtros para recuperá-las posteriormente.
Programemos isto. Criaremos antes de tudo, uma variável à qual, neste caso, chamaremos de "webSession".

Ao colocar este nome, GeneXus já atribui a ela o tipo de dados WebSession, entendendo que certamente seja o que nos interessa, o que neste caso, é efetivamente isso. Caso isto não esteja de acordo, sempre podemos alterar o tipo de dados que se atribua automaticamente pelo

que queremos.
Então, devemos avaliar em que momento queremos que esta variável salve o ou os valores desejados.
Recordemos os eventos principais que temos programados no momento:

- O evento Start, que será executado apenas uma vez quando carrega a página pela primeira vez.

- O evento Refresh, que será disparado após o evento Start e toda vez que for alterado algum filtro que esteja dentro das condições do grid ou for atualizada a página a partir do navegador.

- E o evento Load, que será executado após o evento Refresh, tantas vezes quanto dados sejam carregados em nosso grid.
Este evento será executado N vezes por ter tabela base associada.

Destas opções, claramente a mais conveniente seria no evento Refresh, pois cada vez que alteramos algum dos valores dos filtros, necessariamente será disparado este evento, e é quando estes seriam salvos em nossa variável de sessão, para que possamos recuperá-los ao retornar da transação.
Mas também temos outro evento, que é o evento Click da variável update

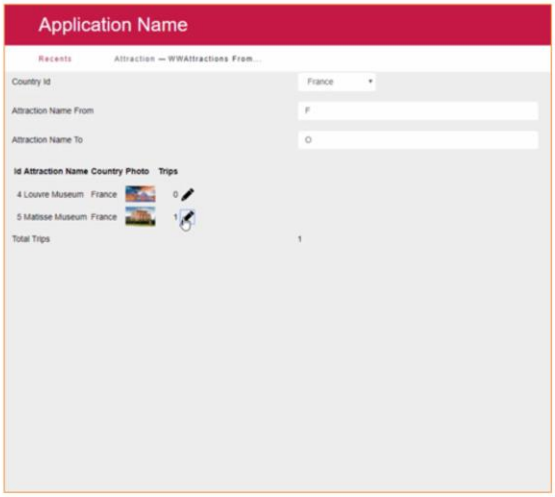
Este evento será disparado imediatamente após clicar na ação atualizar. O que também seria uma boa opção para aqui salvar os valores dos filtros.
Consideramos que é uma boa opção, pois é neste evento onde chamaremos a transação Attraction e, como vimos, é esse o momento exato em que o objeto chamador é destruído e com ele suas variáveis. Portanto, precisamos justamente antes disto, salvar os valores das variáveis de filtro.

Se fizermos isso no evento Refresh, salvaremos os valores dos filtros toda vez que um dado for alterado em algum deles. Já que por estar nas condições do grid, cada alteração dispara este evento. Mas isto não é necessário, pois talvez na execução atual não seja necessário atualizar as atrações filtradas. E nesse caso, para que salvaríamos esses filtros na sessão, se as variáveis não serão destruídas? Não seria necessário.
No entanto, se fizermos isso no evento relacionado à ação de atualizar, salvaremos estes valores apenas uma vez, quando é essencial fazê-lo, imediatamente antes que sejam destruídos o objeto e suas variáveis.
Bem, vamos fazê-lo no evento click da variável.

WebSession Data Type

```
Event Start
    &update.FromImage(updateIcon)
    &CountryId = &webSession.Get('CountryId').ToNumeric()
    &AttractionNameFrom = &webSession.Get('AttractionNameFrom')
    &AttractionNameTo = &webSession.Get('AttractionNameTo')
Endevent
```

```
Event &update.Click
    &webSession.Set('CountryId', &CountryId.ToString())
    &webSession.Set('AttractionNameFrom', &AttractionNameFrom)
    &webSession.Set('AttractionNameTo', &AttractionNameTo)
    Attraction(trnMode.Update, AttractionId)
Endevent
```



Escrevemos a variável webSession, aplicaremos o método "set" para armazenar valores nela e, como primeiro parâmetro, deveremos inserir uma chave (Key), que deverá ser um valor do tipo character, portanto, devemos inseri-la entre aspas.
Neste caso, lhe atribuiremos o nome CountryId. Essa chave única é a que mais tarde nos servirá para recuperar o valor que salvamos.
Em seguida, devemos atribuir um valor (value), que será o dado que queremos armazenar em memória. Neste caso, nos interessa salvar o valor da variável CountryId, que será o valor do primeiro filtro que temos na tela.

Levar em conta que o valor que inserimos também deverá ser do tipo Character, portanto, sendo a variável CountryId do tipo numérico, devemos convertê-la para o tipo Character. Conseguimos isto aplicando o método ToString à variável.

Em seguida, fazemos o mesmo para os outros dois filtros, AttractionNameFrom e AttractionNameTo.

Executamos a aplicação novamente.

Vamos inserir valores em nossos filtros.
Lembremos que toda vez que alteramos o valor de algum dos filtros, é disparado o evento Refresh e, em seguida, o Load para cada registro que é carregado no grid.

Em seguida, selecionaremos na ação atualizar de uma atração.

Nesse momento, será disparado o evento &Update.Click, no qual programamos salvar os valores de nossos filtros na variável &webSession. Para chamar posteriormente a transação Attraction. Que como vimos, será o momento em que são destruídos nosso objeto Web Panel e suas variáveis.

Ao alterar algum dado desta atração e confirmar. Aplicará o comando return e, como neste caso equivale a chamar o Web Panel pela primeira vez, novamente executará os eventos Start, Refresh e Load para cada valor a ser carregado no grid.
Bem, agora o que precisaremos é ser capazes de recuperar estes valores que salvamos na variável webSession.

Qual você considera que seria o melhor momento para recuperar estes valores?
Vamos revisar novamente os eventos que temos e avaliar:
No evento Start?
No evento Refresh?
No evento Load?
No evento &Update.Click?

Claramente, o único que nos servirá aqui será o evento Start. Já que, como vimos, este evento é executado apenas uma vez quando carrega o Web Panel pela primeira vez, e como quando retornamos da transação, é o equivalente a chamar pela primeira vez nosso Web Panel, é aí justamente quando necessitaremos recuperar esses valores,

Como vimos, para ser possível atribuir uma chave e valor à variável webSession, fazemos isso com o método set. Agora precisamos saber como recuperar o valor armazenado lá. Fazemos isto através do método Get, para o qual deveremos indicar a chave (key) do valor que queremos recuperar.

Então, para cada variável que utilizamos como filtro, aplicaremos o método Get, passando por parâmetro a chave correspondente a cada um.
Faremos isso primeiro para a variável CountryId passo a passo,
Inserimos a variável CountryId e lhe atribuímos o valor do método Get da variável webSession, passando por parâmetro a chave, ou seja 'CountryId'.

Lembremos do que comentamos anteriormente, as variáveis do tipo Web Session guardam apenas dados do tipo Character. Sendo a variável CountryId do tipo numérico, para atribuir o valor da variável WebSession, que será do tipo Character, deveremos converter essa informação em numérico, isto obtemos com o método ToNumeric().

Em seguida, fazemos o mesmo procedimento para as variáveis AttractionNameFrom e AttractionNameTo.

Executamos novamente a aplicação e testamos agora seu comportamento.
Filtraremos todas as atrações da China compreendidas entre A e M.

Neste caso, nos aparecerá apenas uma atração.
Selecionaremos a ação de atualizar sobre a mesma, e lembremos de que nesse momento, antes de invocar a transação, salvaremos os dados de nossas variáveis de filtro em memória.
Mudaremos a foto, e confirmamos a ação.

Ao retornar ao Web Panel, como vimos, o primeiro evento que será executado será o evento Start. Onde programamos recuperar as informações que armazenamos em nossa variável de sessão, atribuindo esses valores às nossas variáveis de filtro.

Observamos que agora sim, são mantidos os valores inseridos nos filtros conforme desejado.

E, além disso, após o Start, terá sido disparado o Refresh que, por sua vez, terá disparado a carga do grid de acordo com os filtros. Por isso que vemos o grid novamente filtrado, com a foto modificada.

Mas, uma vez que programamos isto, o que acontecerá quando o usuário abrir em seu navegador pela primeira vez este web panel? Porque, nesse caso, não teremos nada salvo e, portanto, nada a recuperar.
Fechemos toda sessão que tenhamos ativa em nosso navegador, e executemos a partir do Genexus novamente a aplicação, fazendo o ciclo completo.
Entramos no Web Panel criado por nós.
E sendo nesta sessão a primeira vez que é executado o Web Panel, como sabemos, a primeira coisa que é disparada é o evento Start.

A variável de sessão procurará pelas chaves inseridas se há informações a serem recuperadas, não haverá, já que ainda não foi salvo nenhum valor na variável &webSession.
Portanto, neste momento, as variáveis utilizadas para nossos filtros permanecerão vazias.
E, como nas condições do grid, declaramos que não aplique nenhum filtro se estas variáveis estiverem vazias, é que todas as atrações são mostradas.

Selecionaremos, por exemplo, para filtrar pelo país França, e vemos que já nesse momento nosso grid aplica o filtro, mostrando apenas as atrações que tenham a França como país.

Como filtro Attraction Name From, inseriremos a letra F e, nesse momento, novamente será disparado o evento Refresh, seguido pelo Load.
Por último, no filtro Attraction Name To, inseriremos a letra O.

Selecionamos a ação de atualizar na atração museu Matisse.
Nesse momento, é disparado o evento &Update.Click, no qual programamos que antes de chamar a transação Attraction, salve os dados das variáveis de filtro em nossa variável de sessão &webSession.
Para isto, utilizamos o método Set da variável de sessão, passando por parâmetro a chave e o valor que queremos guardar. No nosso caso, são três valores os que nos interessa manter em memória. Que são as variáveis &CountryId, &AttractionNameFrom e &AttractionNameTo.

Neste momento, é chamado o objeto Attraction e este se torna ativo. E nosso objeto Web Panel é destruído junto com suas variáveis.
Modificamos algum dado da atração, por exemplo, modificaremos sua fotografia e confirmamos a ação.

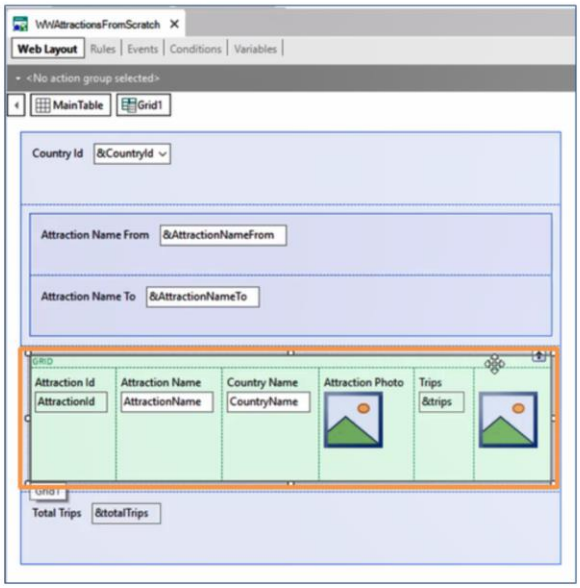
Nesse momento, nos levará de volta ao objeto chamador, ou seja, nosso Web Panel.

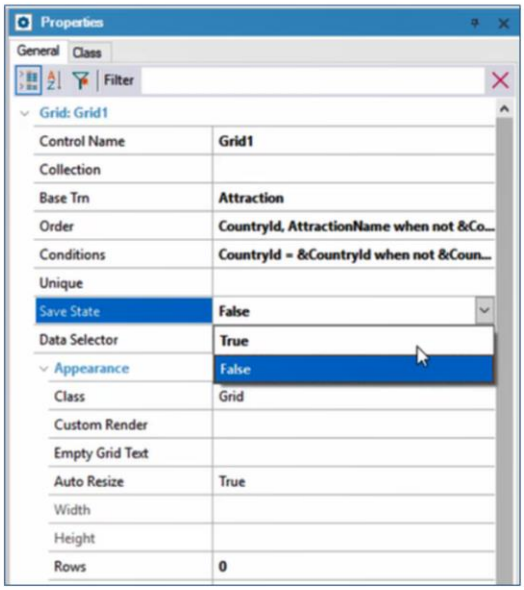
No evento start, serão carregados os valores salvos na variável de sessão. Isto através do método Get e, passando por parâmetro a chave com a qual salvamos cada valor no evento

&Update.Click utilizando o método Set.
Esses dados atribuiremos a cada variável correspondente, no nosso caso, às três variáveis que utilizamos para os filtros. &CountryId, &AttractionNameFrom e &AttractionNameTo..

Depois disso, será executado o evento Refresh e, em seguida, o evento Load para cada registro a ser carregado no grid.
Neste caso, as atrações a serem carregadas no grid, que atendem a essas condições, são duas. Portanto, o Load será executado duas vezes.
E o que vemos na tela é que os valores que havíamos inserido em nossos filtros são mantidos após atualizar um registro.

Save State propery





Desta forma aprendemos a utilizar as variáveis de sessão que nos fornece GeneXus, para a qualquer momento poder guardar informações nela e depois poder recuperar a mesma quando for necessário.

Dada a necessidade frequente como usuários de GeneXus, de ter que salvar o estado em um grid para recuperá-lo depois, por exemplo, como o caso que acabamos de ver, quando chamamos outro objeto e voltamos ao nosso web panel com grid, e queremos que os filtros que inserimos sejam mantidos, é que GeneXus incorporou em suas últimas versões uma propriedade no controle grid chamada "Save State Property". Basicamente o que esta propriedade faz é nos permitir salvar em memória, em uma variável "Web session" todas as informações sobre o estado do grid (a página do grid em que estávamos, os filtros aplicados, etc). Quando esta página é recarregada, automaticamente é restaurada a última configuração conhecida, que é o que fizemos durante este vídeo de forma manual com a variável de sessão, mas será configurando uma propriedade, de forma automática e completamente transparente, pois não veremos o código adicionado nos objetos gerados. Vamos ver em execução.

Antes de mais nada, eliminamos o que fizemos na seção eventos de nosso web panel, deixando-o no estado inicial do vídeo, ou seja, que novamente o estado de nosso grid não seja salvo em lugar nenhum. A partir de nosso objeto web panel, selecionamos o grid e observamos esta nova propriedade que mencionamos, que por padrão nos aparece em false, a mudamos para true e executamos novamente.

Inserimos valores em nossos filtros, selecionamos para modificar as informações de uma de nossas atrações, e como vimos isto chama de um objeto para outro e então quando confirmamos a ação no objeto chamado, retorna para o objeto chamador. Modificamos, confirmamos e ao voltar ao nosso web panel vemos que os filtros são mantidos, recuperando o
estado em que estava antes de chamar a transação Attraction.
Mesmo se tivéssemos uma paginação neste grid, também preserva a posição em que estávamos antes de atualizar o registro.

Temos duas outras maneiras de salvar e recuperar informações de contexto, sobre as quais não entraremos em maiores detalhes.
Uma delas é através do uso dos seguintes métodos do controle grid.
SaveSessionState, que permite salvar o estado do grid. E LoadSessionState, que permitirá recuperar o estado do grid previamente salvo.
Utilizar estes métodos é equivalente ao uso da propriedade save state do grid que acabamos de ver.

A outra é através do uso do objeto externo ClientStorage.

Este objeto é uma API, que se utiliza para armazenar informações de contexto em aplicações para dispositivos móveis. O uso e funcionamento são similares às variáveis WebSession.
Permite armazenar informações com pares chave-valor, de forma local, ou seja, no dispositivo móvel em que estamos, e será possível em seguida acessá-la, mesmo sem conectividade.

Para mais detalhes sobre cada uma destas implementações, convidamos a visitar nossa Wiki.

GeneXus™

training.genexus.com
wiki.genexus.com
training.genexus.com/certifications