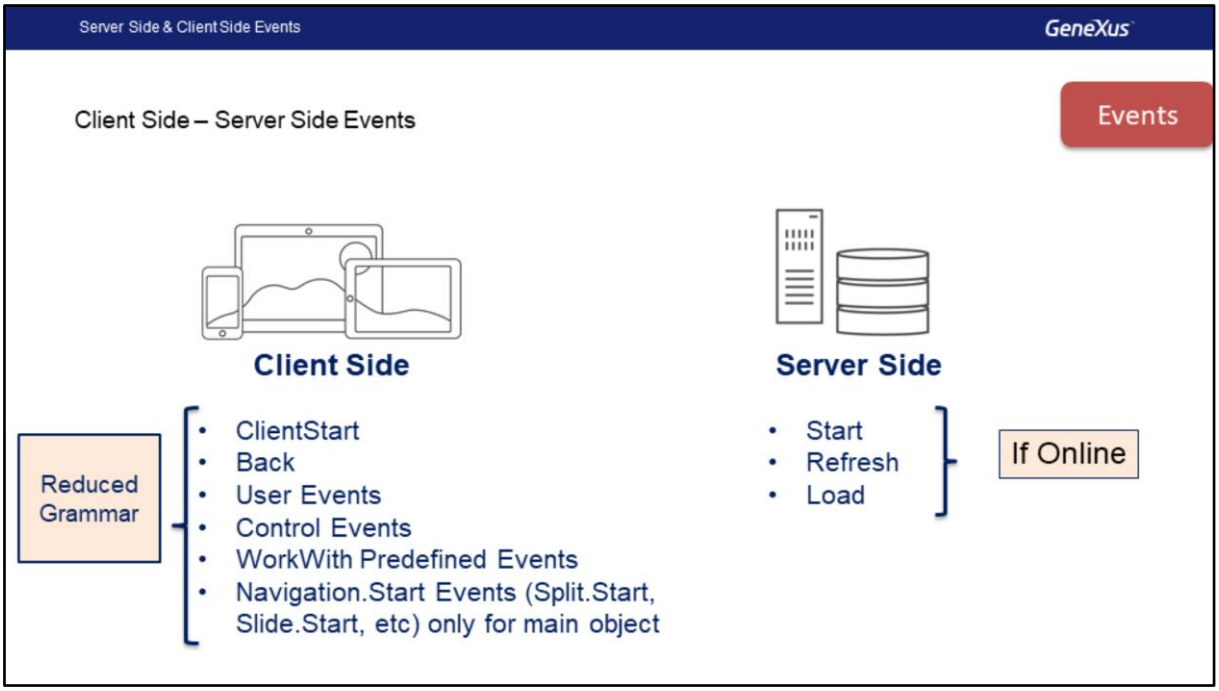
GeneXus™
The power of doing.

Server Side & Client Side Events

Behavior Development

GeneXus 16

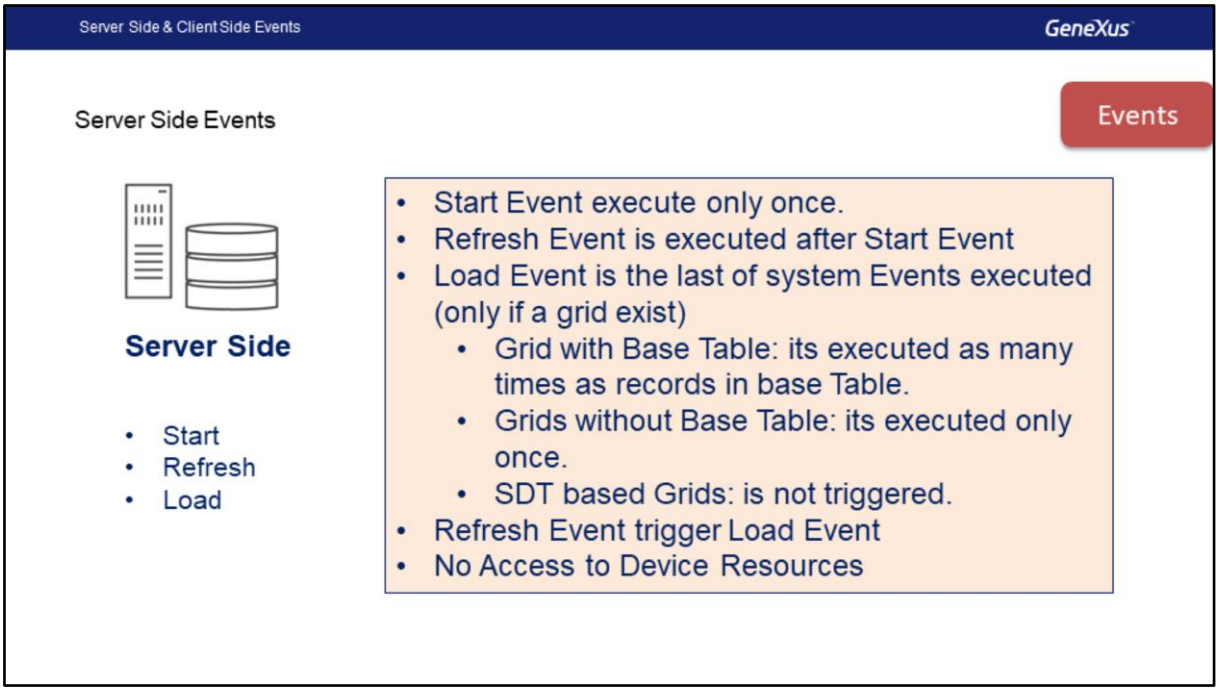


Neste vídeo, começaremos a estudar os eventos que podemos definir no nível dos objetos Smart Devices, no contexto de uma aplicação On-line, destacando os pontos nos quais uma aplicação Off-line difere do que é estudado aqui.

As aplicações para Smart Devices incluem dois tipos de Eventos, os eventos do lado do cliente e eventos do lado do servidor.
Se tiverem experiência no desenvolvimento de aplicações para plataforma Windows ou Web com GeneXus, já estarão familiarizados com os eventos do lado do servidor: Start, Refresh Load.

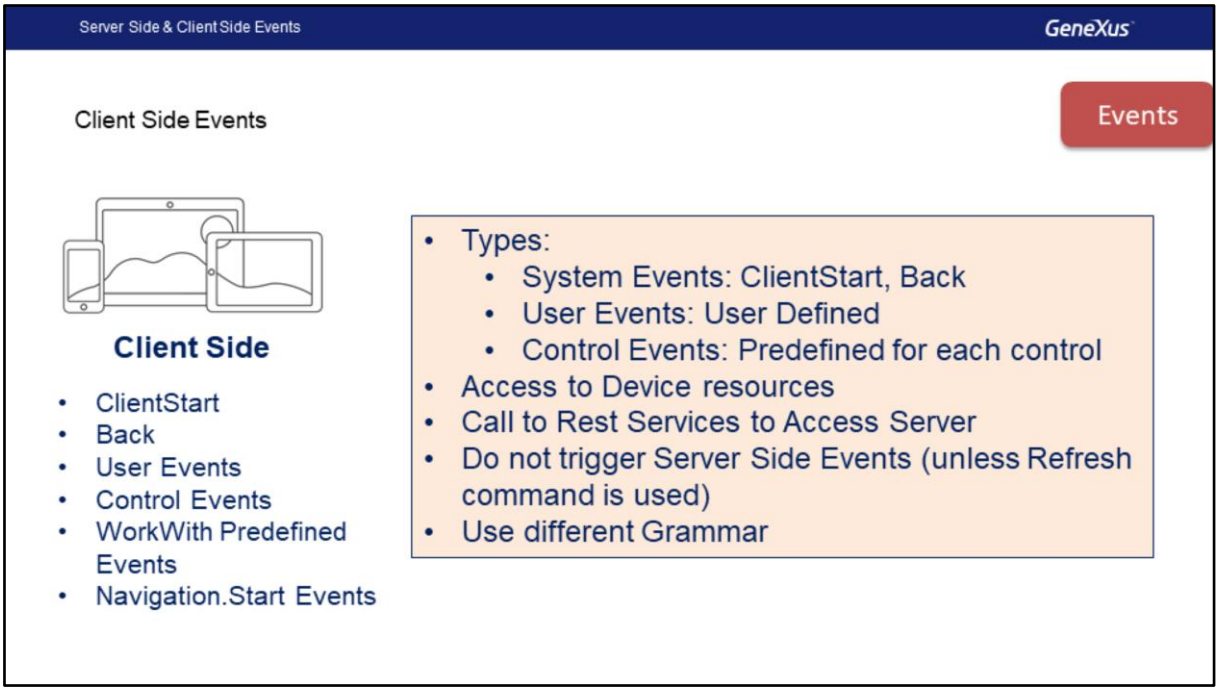
Os eventos do lado do cliente incluem os eventos do sistema ClientStart e Back, os eventos do usuário, eventos associados a controles e alguns predefinidos pelo padrão WorkWith, além dos eventos que estudamos quando vimos os estilos de navegação como Slide.Start, por exemplo.

Esta classificação é importante, pois precisamos saber onde é executado cada evento, já que dependendo disso, os recursos que podemos usar irão variar e também as restrições aplicáveis.
Também existem algumas diferenças em relação à gramática que devemos considerar.



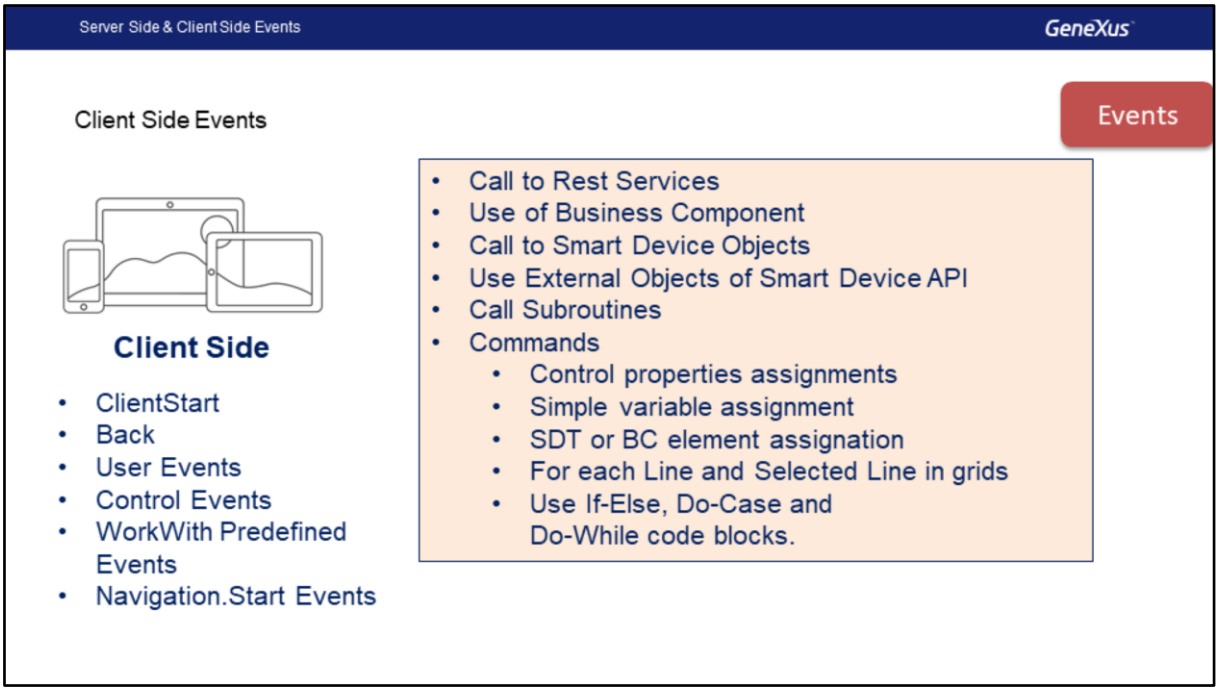
Nestes eventos (Start, Refresh e Load), podemos usar todos os comandos aceitos pelo GeneXus.

- O evento Start será executado apenas uma vez, quando o painel for aberto e não será executado novamente, a menos que saiamos do objeto e voltemos a entrar nele.
- O evento Refresh é executado após o Evento Start, geralmente apenas uma vez, mas pode ser chamado novamente através do comando Refresh, nesse caso, será executado mais de uma vez e será o primeiro evento, pois Start não será mais executado.
- O evento load é o último dos eventos do sistema a ser executado e
    - Se tiver Tabela Base, será executado quantas vezes houver registros na tabela base.
    - Se o grid não tiver tabela base, será executado apenas uma vez.
    - E se o grid for baseado em um SDT, o evento load não será executado.
- Quando for chamado o evento Refresh, ao finalizar, será disparado o evento Load (se aplicável)
- Estes eventos Não têm acesso aos recursos do dispositivo, por exemplo, a câmera, o GPS, etc.



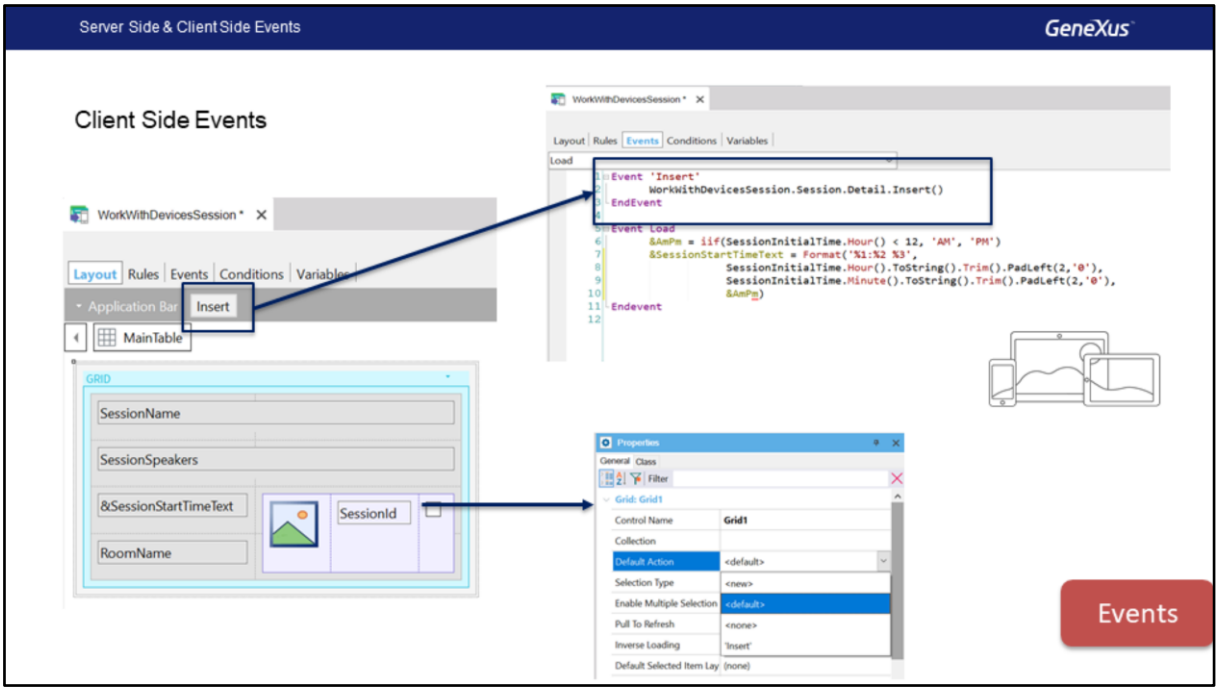
Os eventos do lado do Cliente são a resposta da aplicação à interação do usuário.

- Existem três tipos de eventos do cliente: do sistema, do usuário e eventos de controles.
- Diferentemente dos eventos Start, Refresh e Load, que são executados completamente no servidor, os eventos do lado do cliente têm acesso a todos os recursos de nosso dispositivo.
- O código associado a estes eventos é executado no dispositivo, a menos que seja necessário acessar o servidor (por exemplo, quando deve ser chamado um procedimento, neste caso, o processo sempre será executado no servidor por meio de uma chamada a um serviço Rest, isto é transparente para o desenvolvedor).
- Durante a execução de um evento do lado do cliente, os eventos do sistema não são executados, a menos que sejam requisitados explicitamente através do comando Refresh.
- Como já mencionamos, estes eventos terão uma gramática específica diferente dos eventos do lado do servidor. Discutiremos esse tema em detalhes em outro vídeo.



O que podemos fazer em um evento do lado do cliente?

- Podemos chamar Serviços Rest: automaticamente quando chamamos processos ou Data Providers, estes serão expostos como serviços Rest.
- Usar Business Components para recuperar ou atualizar informações; nesse caso, esses Business Components também serão expostos como um serviço Rest automaticamente.
- Chamar qualquer outro objeto para Smart Devices como nós do WorkWith, Panels ou Menu for Smart Devices.
- Podemos usar os Objetos Externos da Smart Device API
- Podemos chamar sub-rotinas
- E podemos utilizar os seguintes comandos
    - Atribuição de propriedades para controles
    - Atribuição de variáveis simples
    - Usos de variáveis SDT ou Business Component
    - Execução de For Each Line e For each Selected Line em grids
    - Uso dos blocos IF-Else, Do-Case e Do-While

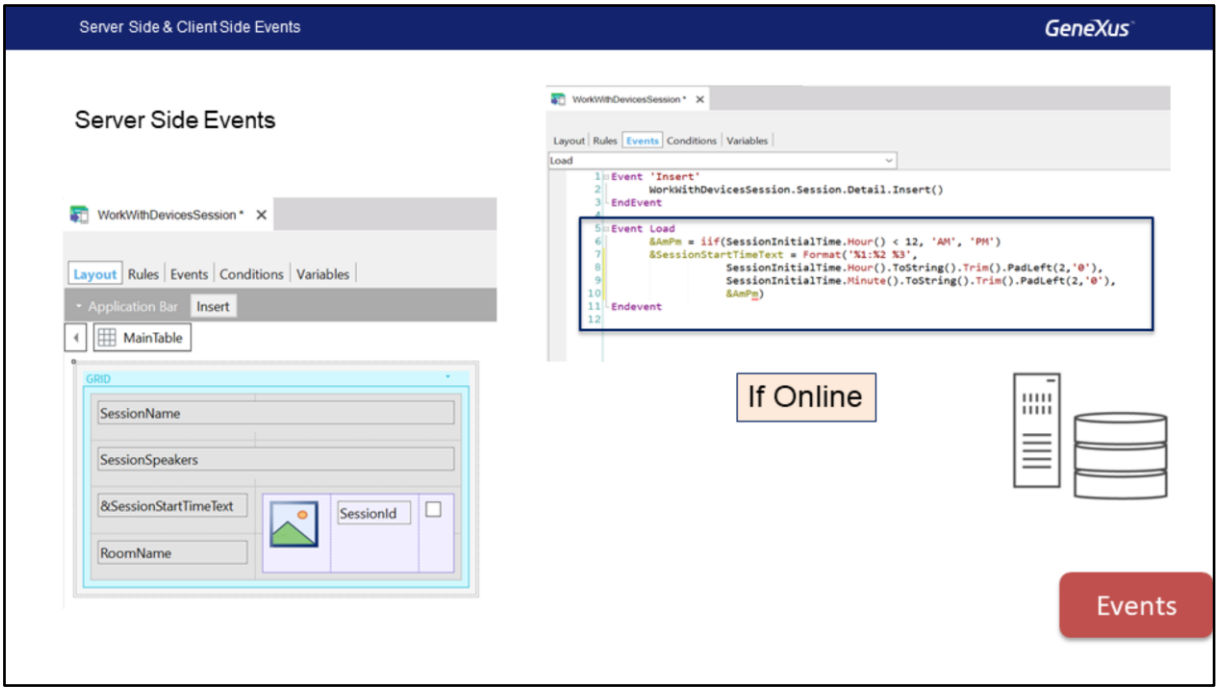


Vejamos alguns exemplos de eventos do lado do Cliente.

Por exemplo, o evento Insert no List é um evento criado pelo padrão.

Outro evento executado no lado do cliente ocorre quando fazemos o TAP em um item do grid, isto dispara um evento que, embora não o vejamos programado nos eventos (ou seja, está implícito por padrão), possui uma Call ao Detail desse WorkWith no modo Display, se quiséssemos, poderíamos alterá-lo e ainda assim continuaria sendo um evento do lado do cliente, pois está associado ao Tap em um controle, neste caso o grid.

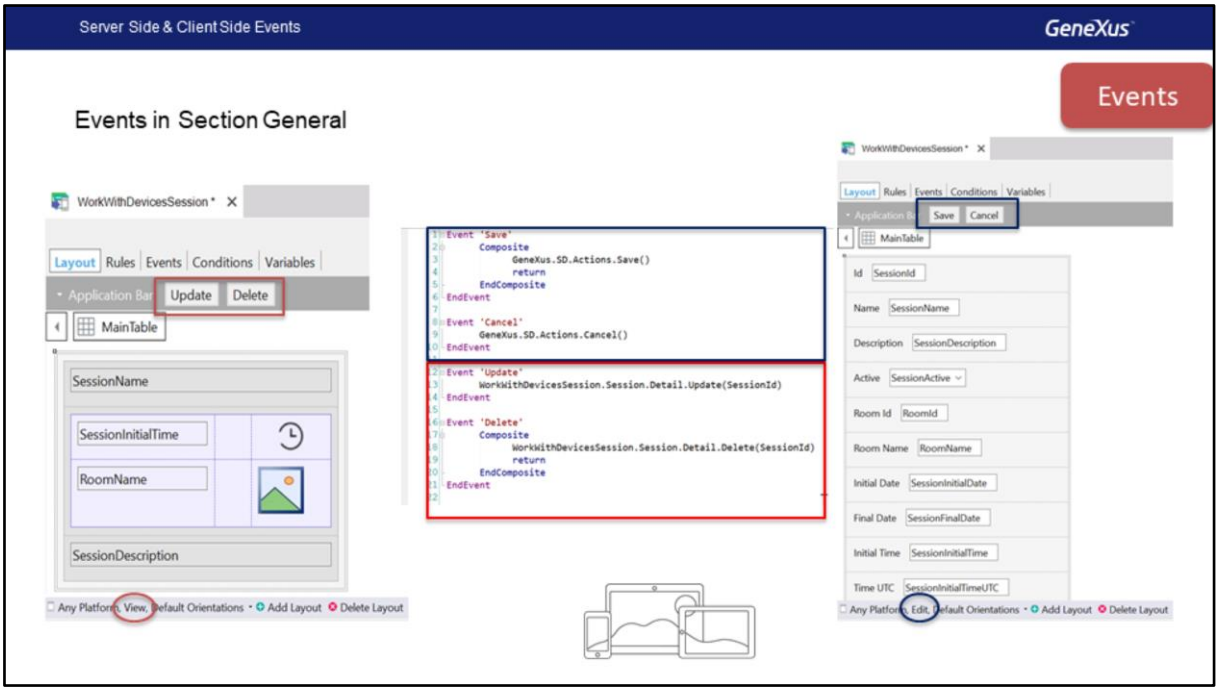
Todos estes eventos são do cliente, mesmo que invoquem objetos que devam chamar serviços (ou data providers) do servidor para retornar os dados ou salvá-los.



Vejamos o que acontece do lado do servidor,

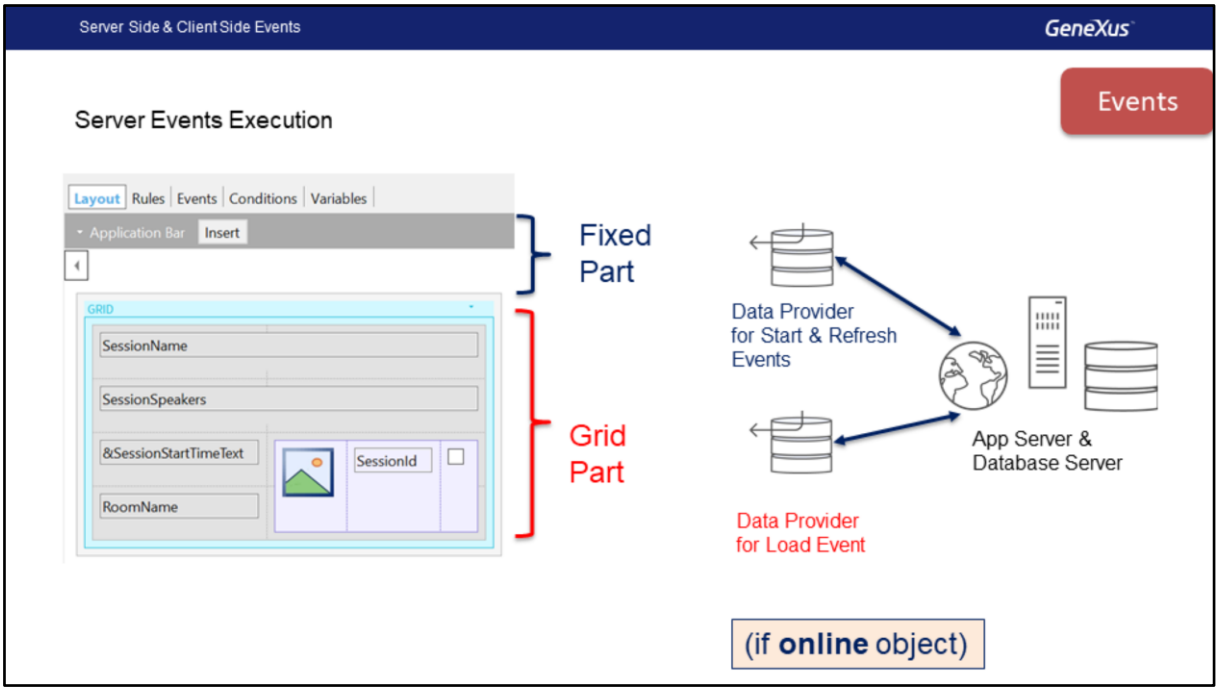
O evento Load, para carregar as linhas no grid, neste exemplo foi programado para que, com base no valor da hora de início, seja carregada uma variável que conterá a hora de início e o indicador de AM ou PM, conforme corresponda.

Este evento será executado no servidor (se o objeto for executado On-line).



Aqui, vemos os layouts criados pelo pattern para a Section (General) e os eventos default. À esquerda, vemos o layout no modo View e à Direita no modo Edit.

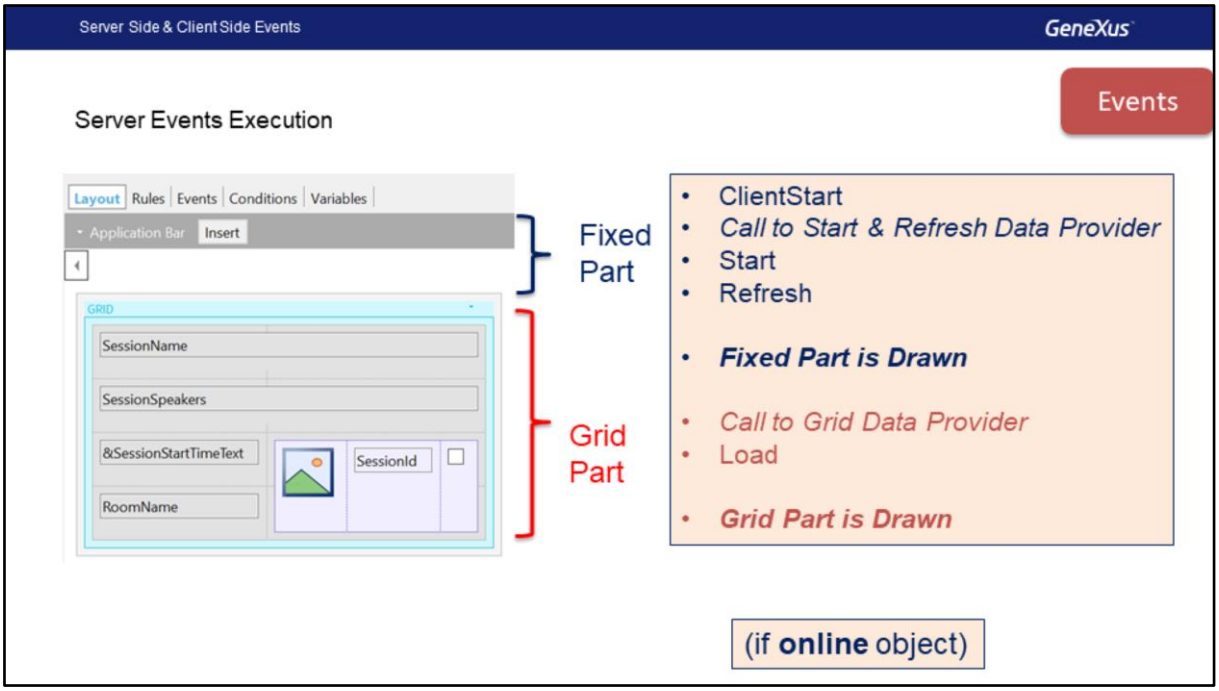
São quatro eventos no total: os dois primeiros Save e Cancel se aplicam ao modo Edit e os outros dois, Update e Delete ao modo View. Todos estes também serão eventos do cliente.



Nos objetos para Smart Devices, podemos identificar duas partes distintas, a primeira parte que chamaremos de parte fixa ou plana conterá tudo aquilo que está no formulário e que não está incluído em nenhum grid, no exemplo que vemos na tela, a parte fixa é composta pela Application Bar com o botão Insert, se tivéssemos atributos fora do grid, eles seriam da parte fixa.
A segunda parte será chamada parte do grid ou variável e, neste caso, será composta pelo grid que contém os dados da Conferência.

Sempre teremos uma parte fixa e também teremos uma parte Variável para cada um do grids que o painel possua, com as quais essa parte poderá ou não aparecer em nossos objetos, dependendo se temos um grid no objeto ou não, neste caso, teremos uma parte Grid.

Para cada parte (fixa e grid) GeneXus gerará automaticamente Data Providers que resolverão o acesso aos dados, estes Data Providers serão invocados com o protocolo Rest, não os veremos na Base de Conhecimento, pois GeneXus será responsável por gerá-los e mantê-los, mas sim, poderemos ver quais dados acessam cada um, como veremos em um momento.



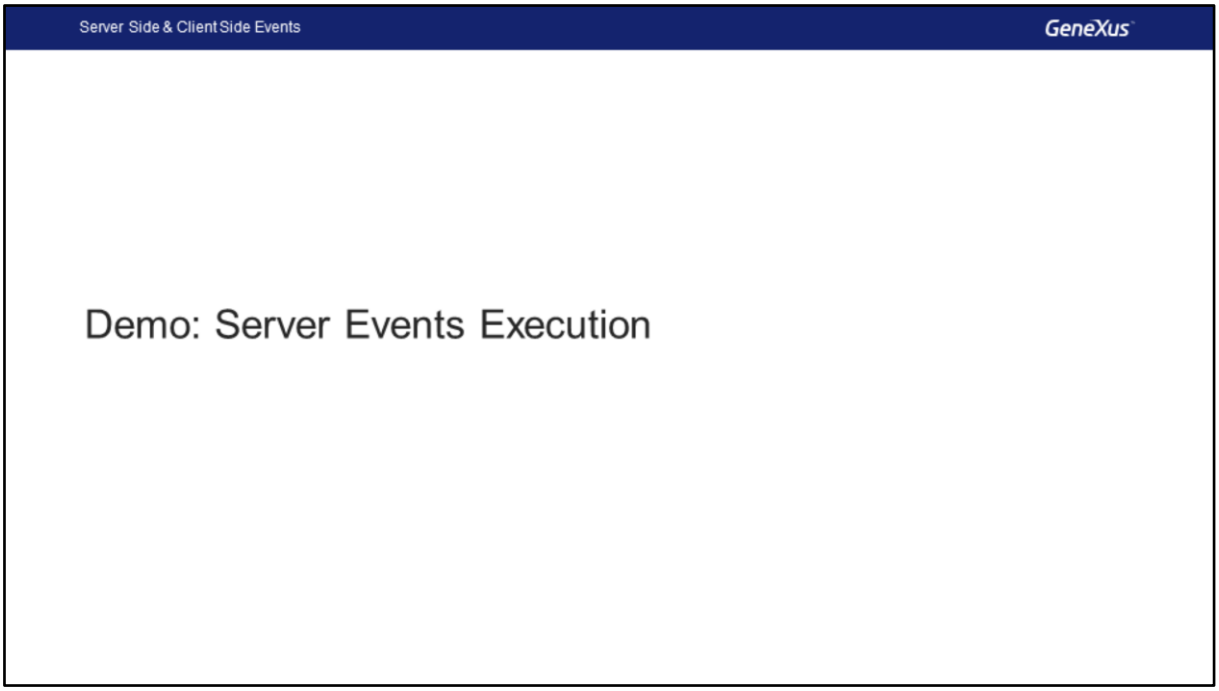
Agora vejamos conceitualmente o que acontece quando executamos pela primeira vez um panel para Smart Devices:

Primeiro, é executado o evento ClientStart, é executado apenas uma vez e da forma como já vimos no dispositivo; em seguida, é executado um data provider através de uma chamada Rest, que resolverá o acesso aos dados necessários para a parte fixa e são recuperados todos os atributos necessários, na sequência, com o resultado retornado pelo serviço, o evento Start será executado e então o evento Refresh no lado do servidor, então será desenhada a parte fixa do layout.

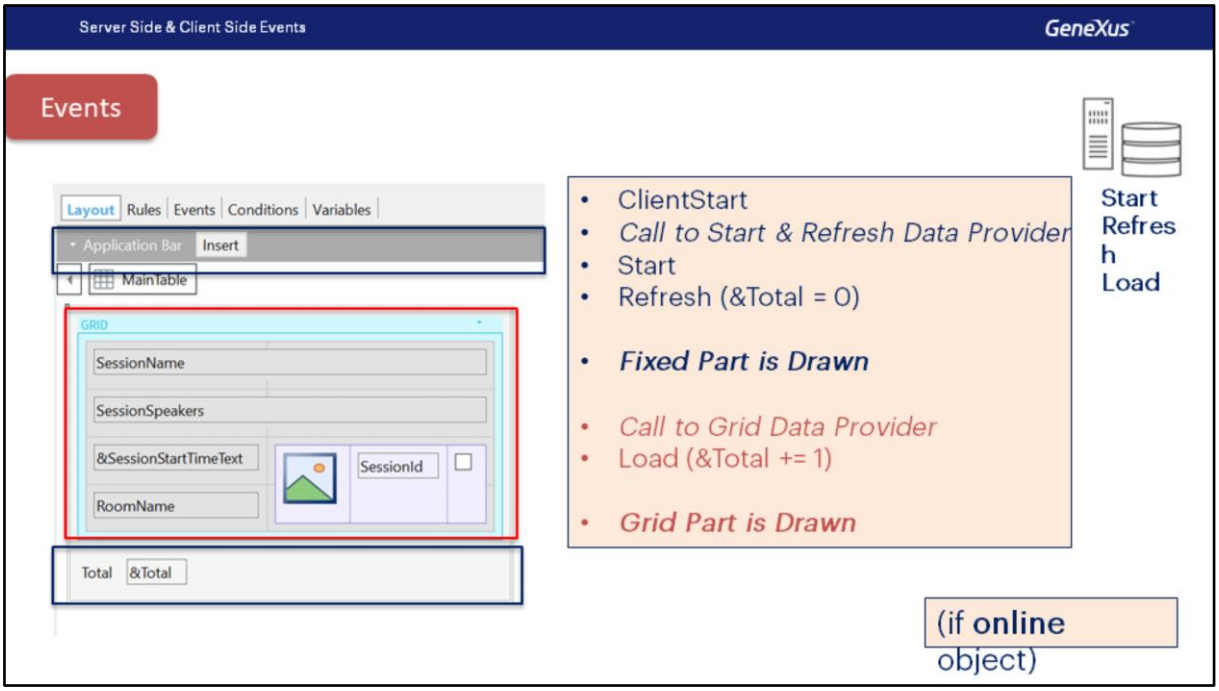
Em seguida, se executa um segundo Data Provider também chamado como um serviço rest, que resolverá a recuperação dos dados requeridos pelo grid e, na sequência, o evento Load será executado. Este evento será executado N vezes quando o grid tiver tabela base, uma vez para cada registro.

Neste momento, terminará de desenhar o layout com a parte do grid.

A particularidade de que a tela seja desenhada em dois momentos diferentes tem suas consequências, como veremos a seguir.



Vejamos um exemplo em GeneXus

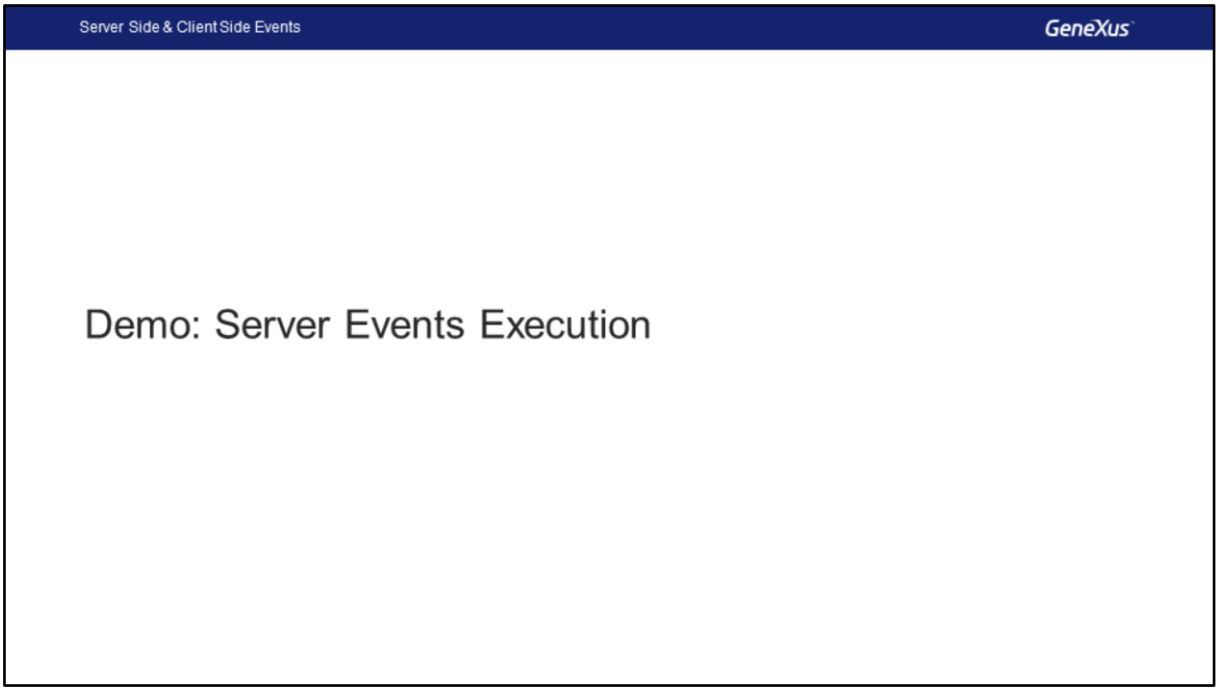


Aqui podemos ver esquematicamente o que fizemos.

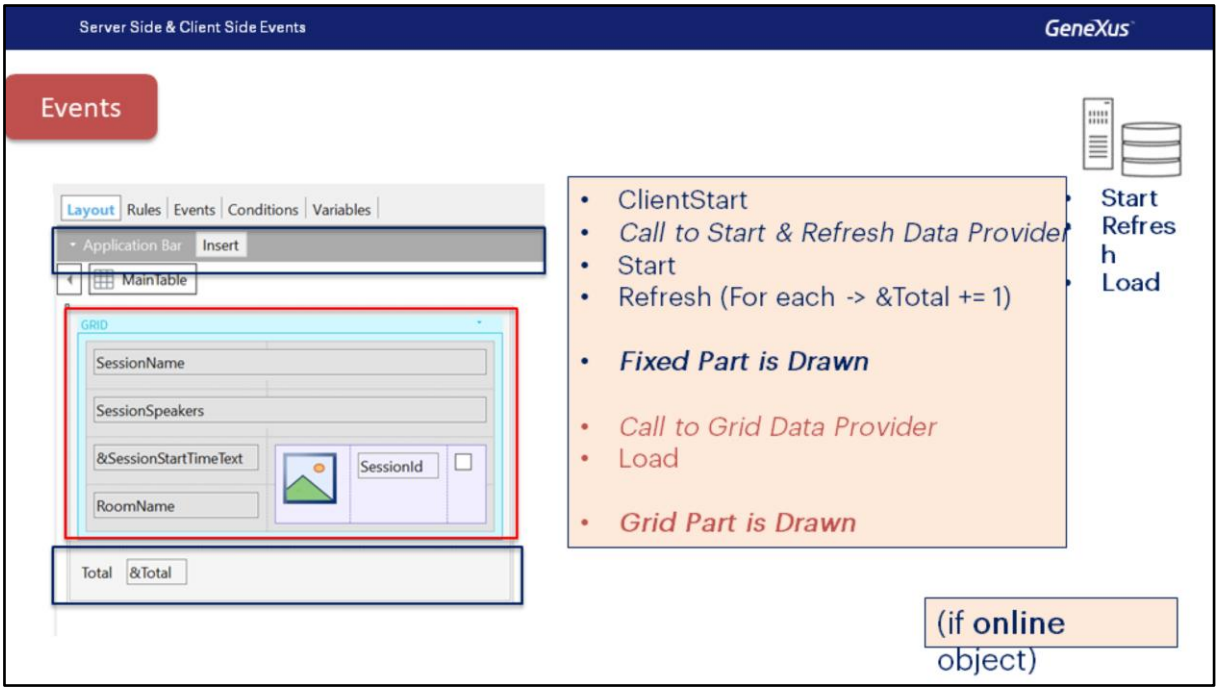
Se lembrarmos do que vimos antes, ao abrir o painel, o Evento ClientStart foi executado. Neste momento, foi chamado o Data Provider da parte fixa que, no nosso caso, não recupera nada do servidor, então foram executados os eventos Start e, em seguida, Refresh onde foi inicializada a variável &Total, na sequência a parte fixa foi desenhada; observe que a variável &Total neste ponto valia zero.

Em seguida, foi chamado o Data Provider associado ao Grid e foi executado o evento Load, aqui incrementamos o valor de &Total, mas a variável &Total já estava desenhada no Layout com valor zero. Então é desenhado o grid.

Observe que esse é o motivo do problema, a ordem na qual são desenhadas as diferentes partes do objeto, não é o fato de que os eventos Refresh e Load estejam separados em dois programas (dois data providers independentes).



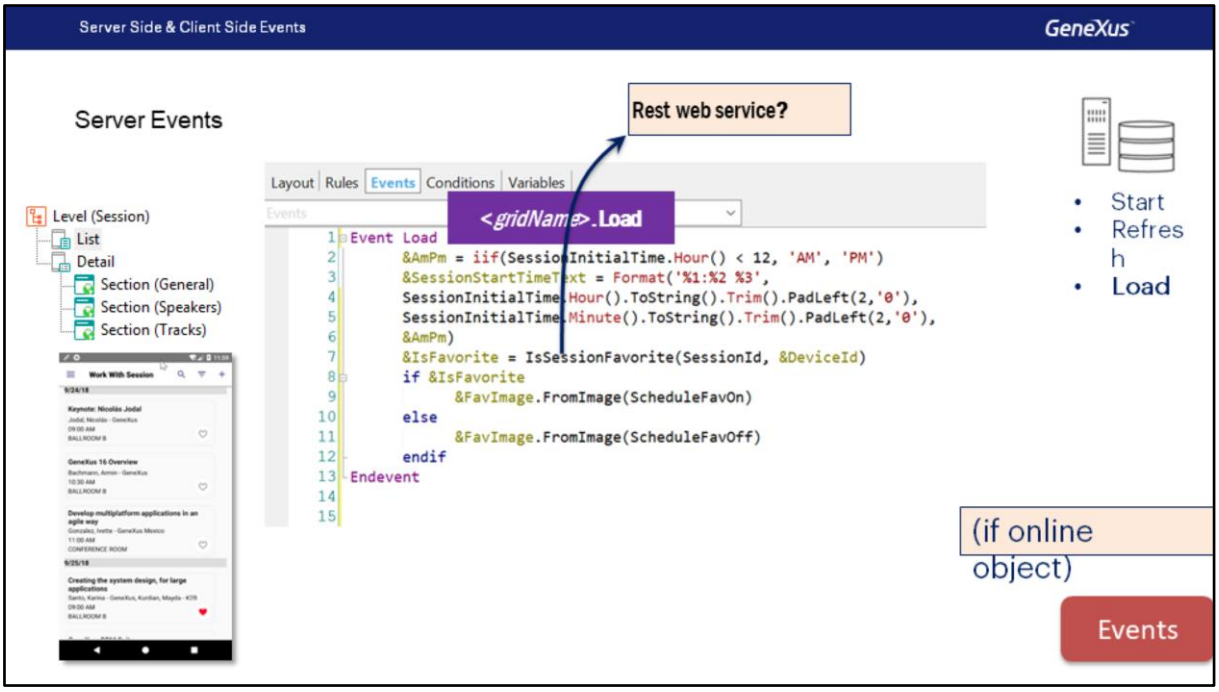
Voltemos ao GeneXus.



Vamos ver o que aconteceu agora:

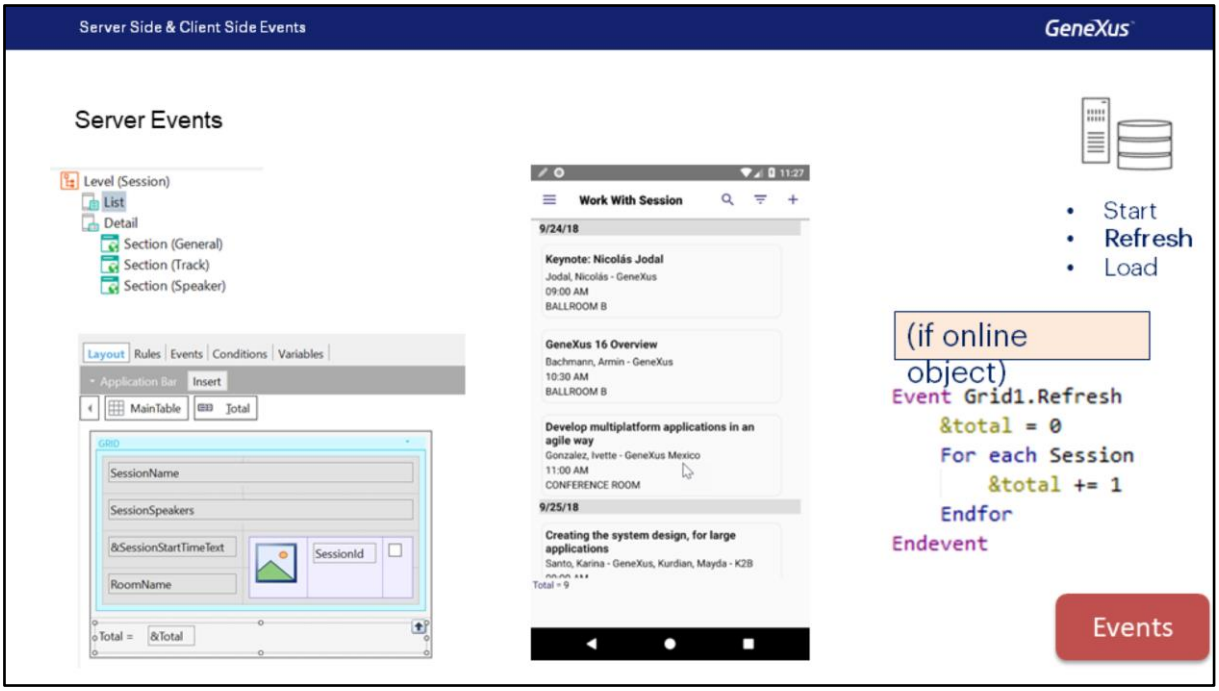
Foi executado ClientStart, neste momento, foi chamado o Data Provider que resolveu a parte fixa que recuperou o resultado do For Each na tabela Session onde fez o Count, depois foi executado Start e então Refresh onde se incrementou a variável &Total, isso é otimizado e diretamente terá o resultado do count da tabela porque foi retornado o Data Provider que vimos, em seguida a parte fixa foi desenhada, observe que agora a variável &Total vale nove.

O restante permaneceu o mesmo, foi chamado o Data Provider associado ao Grid e foi executado o evento Load n vezes. Então o grid foi desenhado.



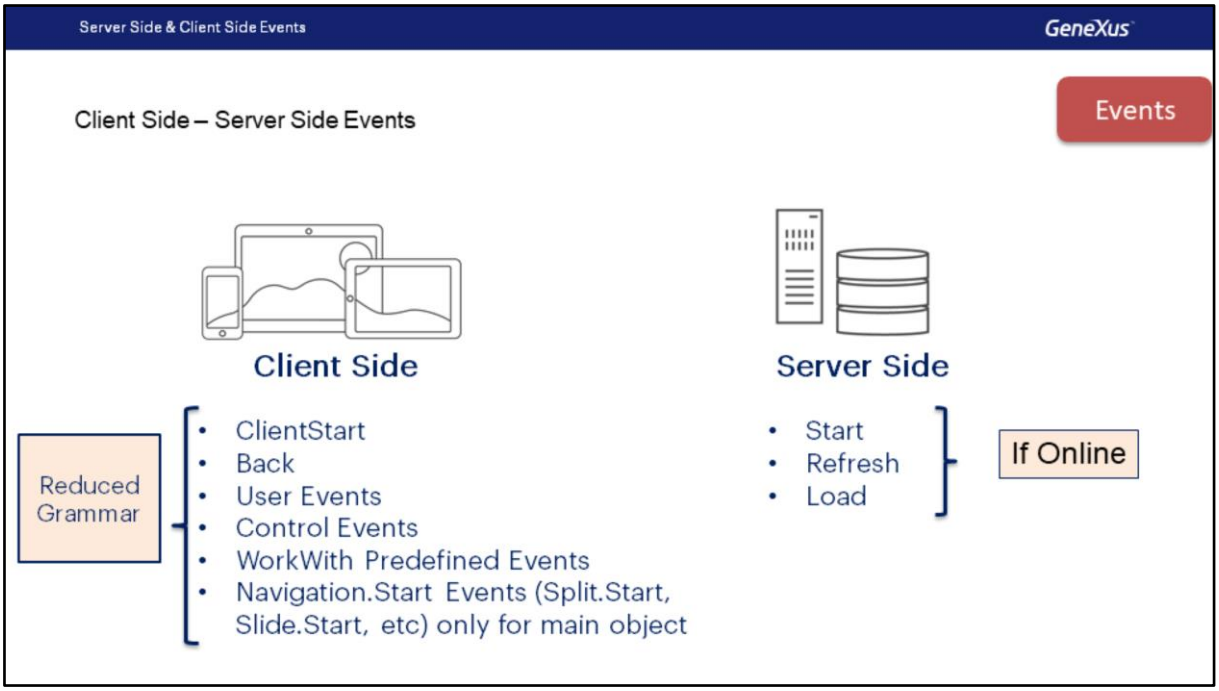
O evento **Load** do sistema não requer nenhuma consideração especial, pois é análogo ao Load de um Web Panel. Como é possível incluir vários grids com tabela base no mesmo layout, como nos web panel quando temos mais de um grid, será necessário especificar o Load de qual grid estamos programando, escrevendo o nome do grid *ponto* **Load**.

Aqui, mostramos o exemplo do Load para carregar os itens do grid do List do WorkWithDevicesSession.
Este evento é executado no servidor, portanto, tem à disposição todos os atributos da tabela estendida para serem utilizados.
Observe que o procedimento IsSessionFavorite não precisa estar exposto como um serviço Rest, pois está sendo chamado a partir do próprio servidor.



O evento Refresh, do sistema, como o Load, não requer nenhuma consideração especial, pois é análogo ao Refresh de um Web Panel.
Também é válido, o mesmo que com web panels quando temos mais de um grid, ou seja, será necessário, nesse caso, especificar o Refresh de qual grid estamos programando, escrevendo <*gridName*>.**Refresh**.

Este evento, como o Load, é executado no servidor, portanto, tem à disposição todos os atributos da tabela estendida, para serem utilizados.



Vimos então os eventos que são executados no cliente e aqueles que são executados no servidor.
É impotente diferenciá-los, uma vez que o que podemos programar nos eventos do cliente segue uma gramática um pouco mais reduzida que o que podemos fazer no servidor, como veremos especificamente em um vídeo separado.

Adiantamos que isto não afetará a implementação de uma aplicação off-line. Ou seja, quanto à gramática, você programará os eventos da mesma maneira se sua aplicação for on-line ou off-line. Nesse sentido, é transparente.

No vídeo seguinte, continuaremos estudando a navegação e determinação da Tabela Base nos objetos para Smart Devices.

Videos	training.genexus.com
Documentation	wiki.genexus.com
Certificactions	training.genexus.com/certifications