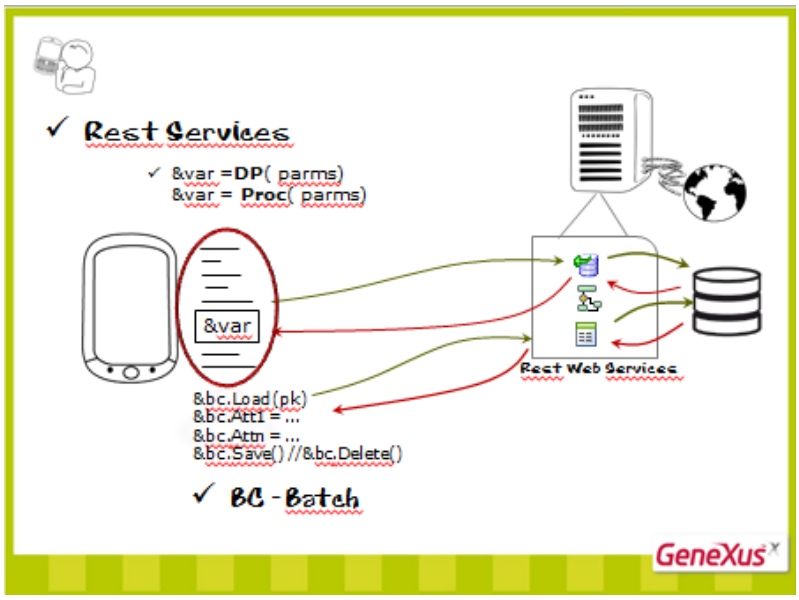
Curso GeneXus - Eventos em Smart Devices

Há eventos cujo código é executado no servidor e eventos cujo código é executado no cliente (ou seja, no dispositivo).

Vamos abordar, agora, os tipos de ações que podem ser escritas dentro de um evento de usuário.

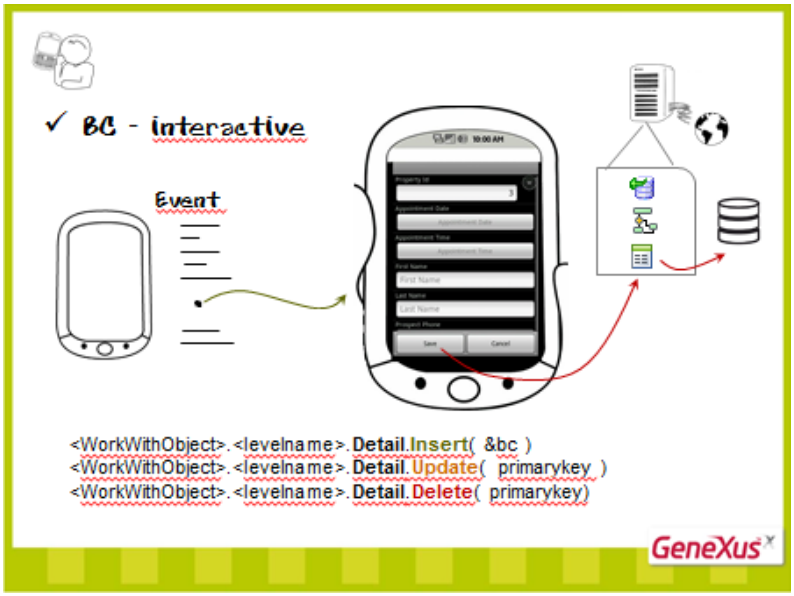


Suponhamos que este é código do evento no cliente. Por outro lado, temos o servidor Web e a base de dados. ¿O que podemos fazer aqui?

Podemos acionar os serviços rest do servidor, como data providers, ou procedimentos que busquem na base de dados e nos devolvam as informações que carregaremos numa variável. Necessariamente, eles devem estar expostos como serviços rest. Não podemos acionar um procedimento interno desde o dispositivo. Por exemplo, corresponde a escrever no Evento:

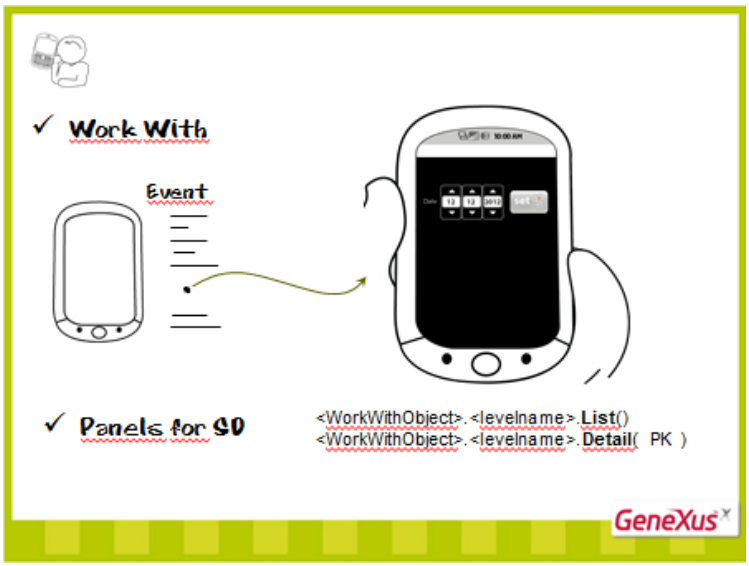
Também podemos querer, dentro de um evento, inserir um novo registro à base de dados sem ter que pedir informações ao usuário. Isto é feito, como em qualquer outra ferramenta GeneXus, com os métodos e propriedades do BC; no entanto, aqui deverá estar exposto também como serviço rest, dado que estamos acionando desde o dispositivo. É o caso de atualização batch.

Em resumo: podemos acionar o serviços Rest: Data Providers, Procedimentos e Business Components.



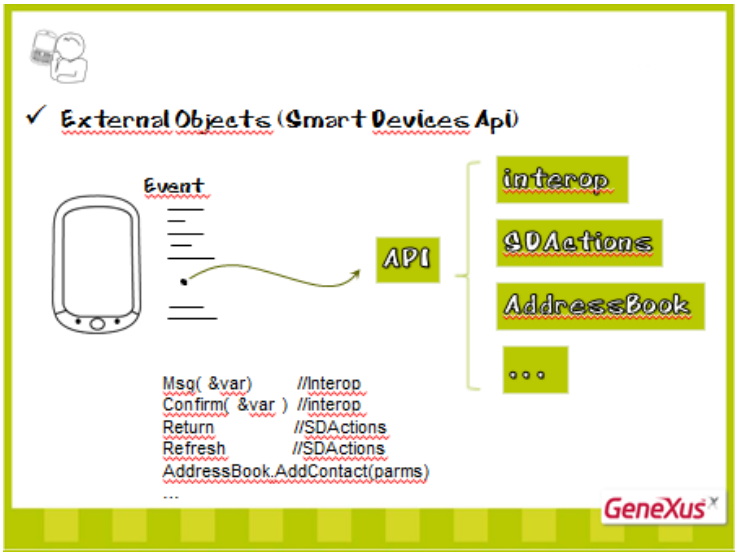
Como foi visto, também podemos acessar a tela de Detail do WorkWith para inserir, atualizar ou deletar. Através da tela acessada, se pedem os dados ao usuário e, em seguida, é realizada a operação correspondente (o que internamente se traduzirá em um acesso ao BC rest).

Portanto, estamos acessando o Detail de um Work with, que utilizará internamente o BC rest cujos dados são pedidos ao usuário através da tela Edit.

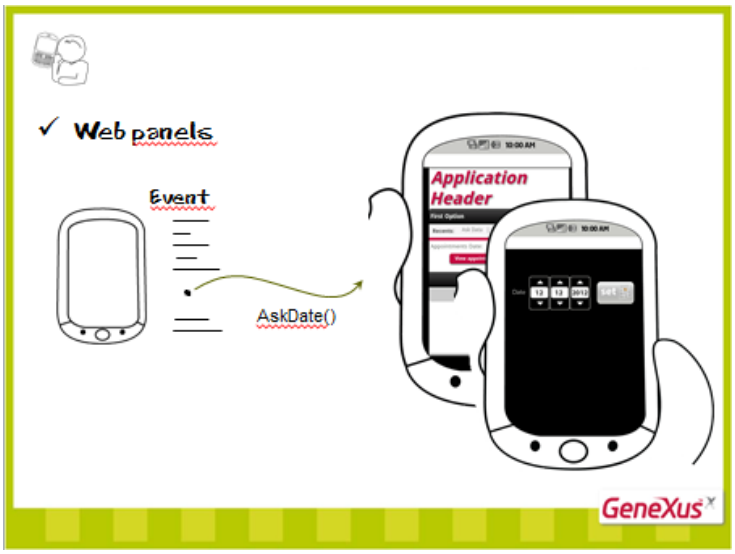


Também poderíamos simplesmente querer acionar o List ou o Detail no modo view.

Assim como ferramentas Panels for Smart Devices, que são telas um pouco mais livres que as dos work with vistos.

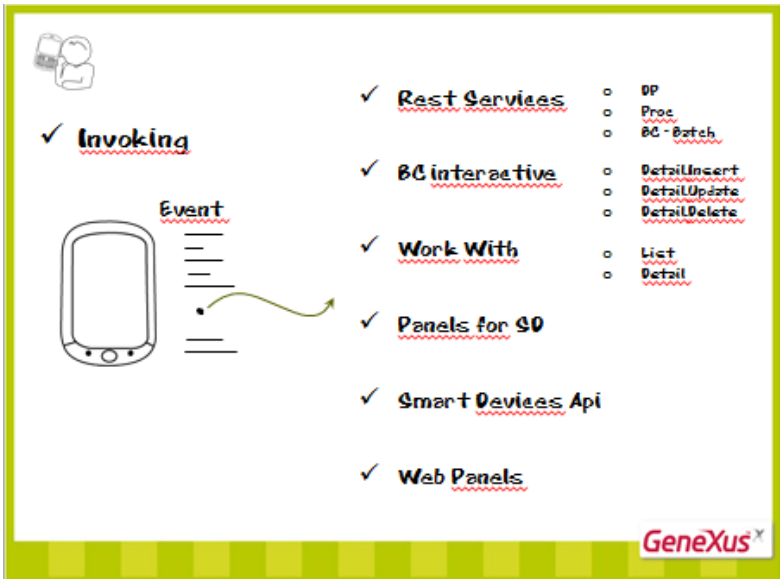


…Ou utilizar algumas das funcionalidades fornecidas pelas apis, como implantar uma mensagem na tela, pedir confirmação ao usuário para continuar, voltar ao acionador, atualizar a tela, incluir um contato em sua agenda de endereços, etc. Destes exemplos, exceto a atualização da tela, todas as demais ações são realizadas sem a necessidade de ir ao servidor para executá-las.



Também um web painel pode ser acionado, ou seja, a um painel do aplicativo web, que implementa e permite incluir dados, mas para esse tipo de aplicativo (como o Panel for Smart Devices faz para aplicativos Smart Devices).

Por exemplo, aqui estamos acionando o web painel chamado AskDate(). Ele será aberto pelo navegador do dispositivo (que terá o símbolo ocultado , para que pareça com o resto do aplicativo). Qual a outra forma de executar uma ferramenta web através de um navegador web?



Até aqui vimos todas as formas de acionar possíveis:

Serviços rest (Data providers, procedimentos e Business Components)

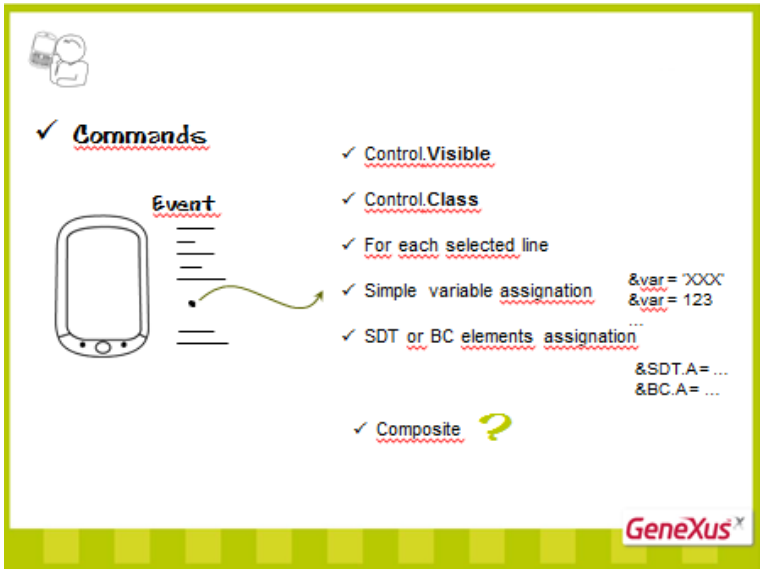
Telas de Detail do Work With no modo Edit, para Insert, Update ou Delete interativamente.

Telas do Work With para visualizar informações (tanto List como Detail no modo View)

Painéis para Smart Devices, que estudaremos em outro vídeo.

Qualquer apis para Smart Devices fornecidas por GeneXus, através de seus métodos.

E Web Panels (este é o único caso de ferramenta do servidor que não é acionada como serviço rest. A razão é simples: na verdade, é um navegador que está sendo executado de maneira transparente)



Mas, fora os comandos de acionamento que acabamos de ver, temos estes outros, que nos possibilitam:

Fazer visível ou invisível um controle de tela.

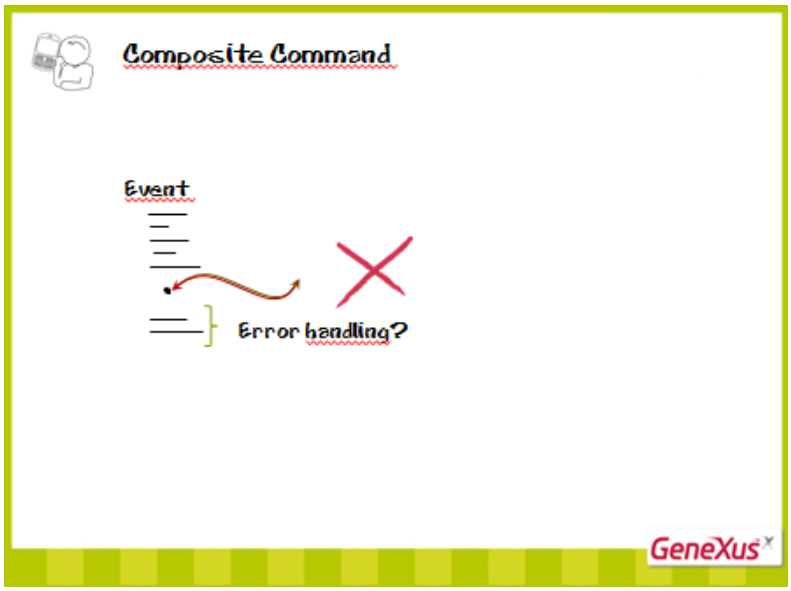
Mudar a classe de um controle de tela.

Percorrer todas as linhas de um grid selecionadas pelo usuário antes de executar o Evento.

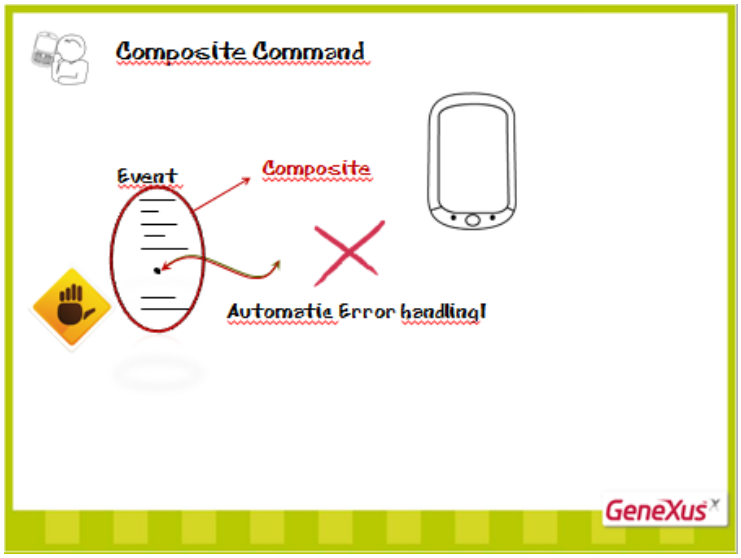
Dar valores a variáveis de tipos de dados simples.

E dar valores a variáveis de tipos de dados compostos (estruturados ou Business Components).

E, por último, temos o comando Composite. ¿Para que ele serve?

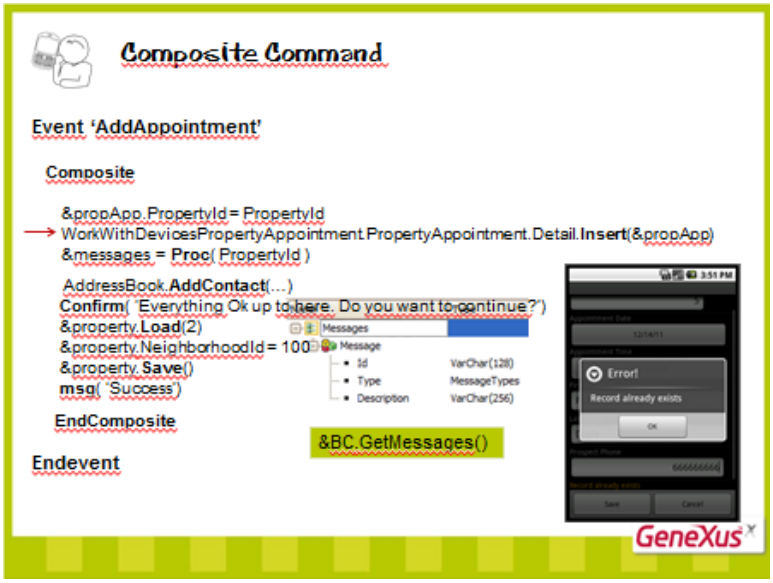


Nos aplicativos Web GeneXus, quando, dentro de um evento que está sendo executado, uma ferramenta chamada produz um erro, a execução não é interrompida; segue-se para a sentença seguinte e o desenvolvedor, dentro do próprio código do evento, deve encarregar-se de solucionar os erros e programar as ações a serem tomadas.



Se queremos outro comportamento, nos casos em que um erro numa sequência de chamadas detenha a execução, para que se solucionem os erros **automaticamente**, implantando-os na tela sem a necessidade de escrever nenhuma programação, precisamos de um comando especial que o especifique. Este comando é o composite.

Está implementado somente em Smart Devices e é obrigatório neles, sempre que for realizada mais de uma ação dentro do mesmo evento.



Neste exemplo, vemos uma sucessão de comandos, alguns dos quais são chamadas.

Se a primeira chamada falhar, por exemplo, por chave duplicada, então a mensagem de erro é mostrada como vemos, e se **detém** a execução (não se executa a chamada seguinte ao procedimento e nem nada do que segue).

Se não for o caso, então esta ação é realizada. No procedimento, se especificarmos como parâmetro de saída uma variável do tipo de dados, o SDT pré-definido Messages (que devolve automaticamente um BC quando executamos seu método GetMessages), e a carregamos dentro do procedimento com as mensagens de erro ou avisos que convenham, ao retornar da execução do procedimento esta variável é inspecionada automaticamente e, em caso de erro, se detém a execução e desaparecem as mensagens da tela.

Caso contrário, é executada a chamada seguinte, e assim sucessivamente.

Acabamos de estudar os eventos cujo código é executado no dispositivo. Agora veremos os eventos que são executados no servidor.

São eventos do sistema: Start, Refresh e Load.

Para introduzi-los, vamos pensar no seguinte exemplo:

Queremos mostrar, no list de imóveis, um check box que esteja marcado para as propriedades que tenham sido muito visitadas, e uma imagem que indicará que a propriedade foi incluída recentemente no sistema.

{demo}

Para isso, no List do Work With de Property, incluímos uma variável booleana…

E a incluímos no Layout…

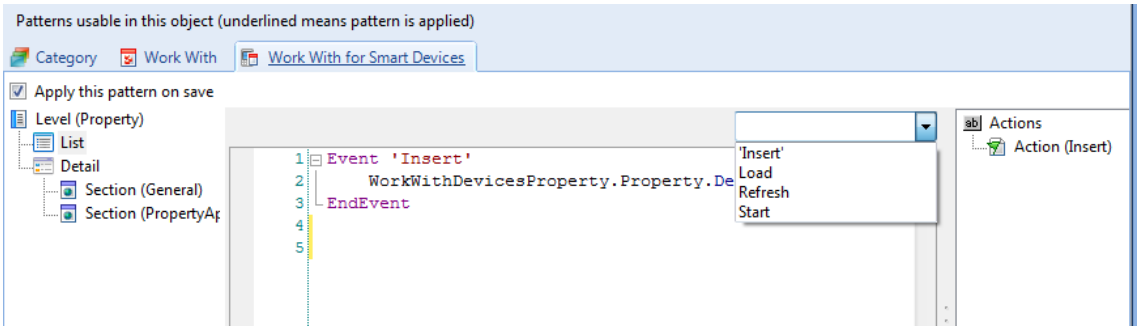
Junto com um controle imagem, ao que chamamos Image1, para poder fazer referência a ele logo.

Criamos um procedimento que recebe como parâmetro o identificador de propriedade e devolve um valor booleano, resultado da avaliação da quantidade de visitas que essa propriedade tem agendada, que é maior que 2. Aqui colocaríamos o valor que dispuséssemos, que marcaria a diferença entre “muito visitada” e o contrário. Para facilitar o testing, colocamos este valor baixo. Assim, se tem mais de 2 visitas, carrega True na variável e, caso contrário, False.

Com isso, tudo o que nos resta é programar a consulta à tabela de imóveis para recuperar cada linha que será carregada no grid; também será carregado o valor da variável {assinalar} com o resultado do procedimento, e será marcado se a imagem que indica se a propriedade é nova, se deve ser mostrada ou não.

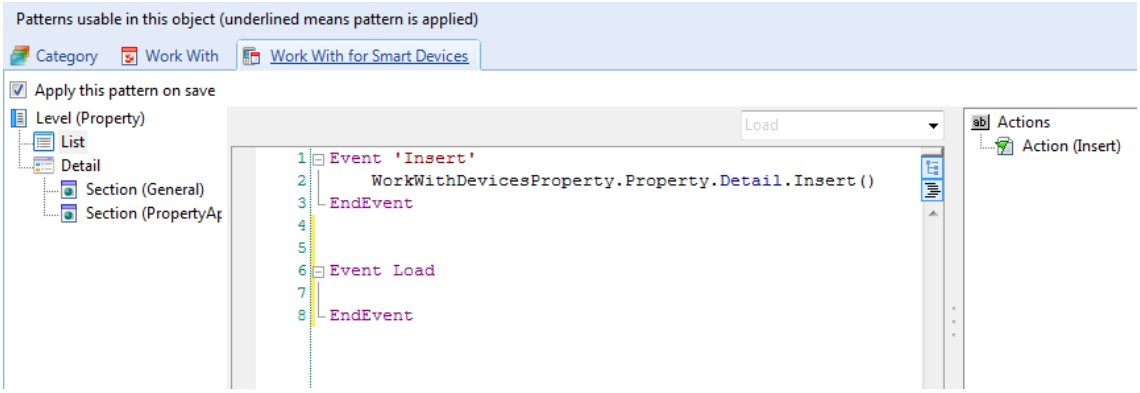
Mas, onde colocamos este código? No evento do sistema Load, que consulta a base de dados, recuperando os registros a serem carregados como linhas do grid.

Aqui podemos ver todos os eventos até agora definidos para o List.



Entre eles vemos os três eventos do sistema que foram mencionados.

Escolhemos o Load. {demonstrar}



E escrevemos:

Apertamos o botão F5.

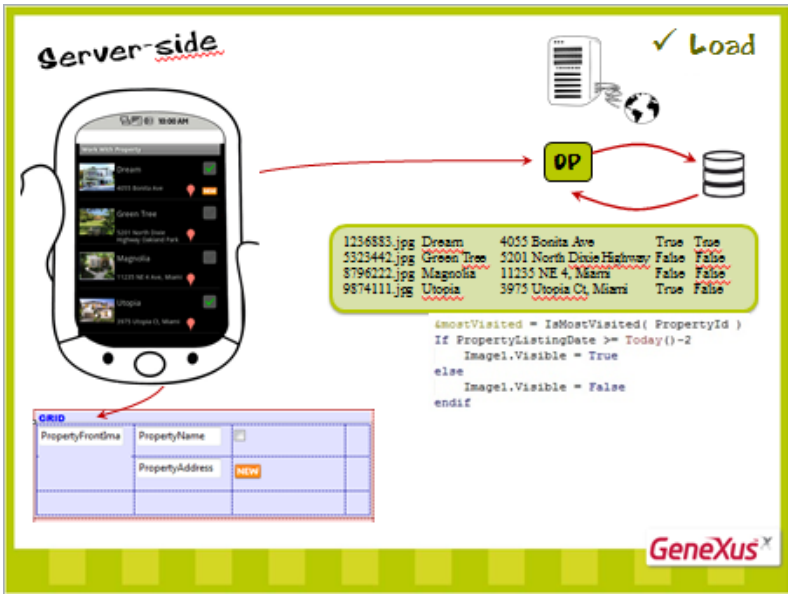
Vemos que Dream está marcada, indicando que tem mais de 2 visitas agendadas. Se vamos consultá-las… tem 3. Além disso, sua data de entrada no sistema é 28 de março de 2012. Sabendo que hoje é dia 29 e que indicamos que é nova se foi incluída nos últimos dois dias, então…está aparecendo a imagem.

Green Tree foi incluída no dia 12 de setembro do ano passado, e tem 1 visita, por isso… nem foi muito visitada e nem é nova.

Magnólia é do dia 26 de junho do ano passado e não tem visitas agendadas, por isso…

Por último, Utopía está marcada como muito visitada. Portanto, temos que encontrar mais 2 visitas, e sua data de entrada no sistema deve ser anterior a 27 de março de 2012...

Como vemos, está sendo cumprido…



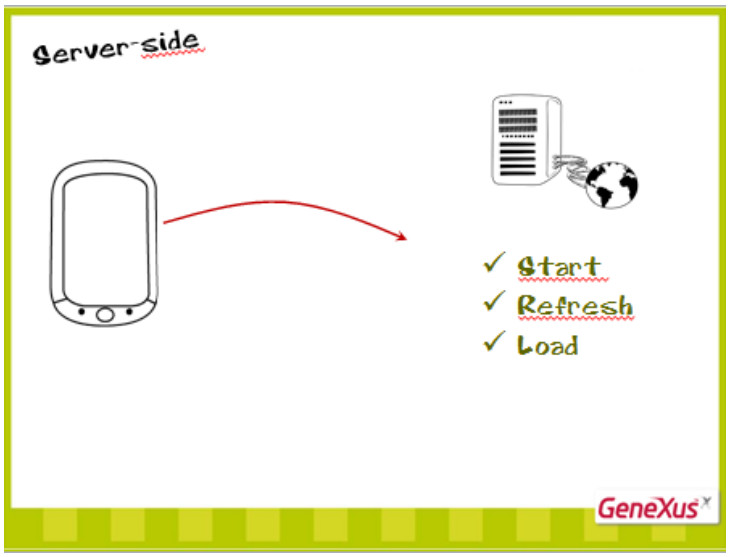
Em suma, para se ter um grid com atributos em uma tela, implicitamente estamos dizendo que você deve acessar à tabela do banco de dados correspondente a esses atributos (e eventualmente sua extensão) e todos os registros que satisfazem às condições, e retornar ao dispositivo de uma coleção com os valores pedidos.

Quem realiza o trabalho de recuperar os dados e enviar-los ao dispositivo, não é nada mais nada menos que um Data Provider Rest, implicitamente criado e gerado por GeneXus, de modo transparente para nós.

Este Data Provider, para cada registro da tabela recuperado para ser incluído na coleção devolvida, executa o código do evento Load. Desta maneira são carregados, para cada imóvel,

os valores True ou False tanto da variável mostVisited, como da propriedade Visível do controle Imagem.

Este código é executado, portanto, no servidor, pelo qual podemos utilizar os comandos que usamos habitualmente nos aplicativos web. Por exemplo, observamos que aqui estamos acionando um procedimento IsMostVisited, que é interno.



O Start será executado também no Server, mas somente quando se executa a tela pela primeira vez. Ou seja, a primeira vez que essa tela do work with é aberta.

O Refresh, por outro lado, será executado toda vez que deva atualizar a tela. Logo após o Refresh, será executado o Load (a carga do grid).

Com isso, completamos a exposição que queríamos realizar sobre os eventos das distintas telas de um Work With for Smart Devices. Tudo o que foi dito vale também para os eventos de Panels for Smart Devices.

Quer implementar painéis que permitem pedir dados ao usuário, mostrar dados não necessariamente provenientes de uma ou várias tabelas da base de dados e dar mais flexibilidade a suas telas?

Continua.