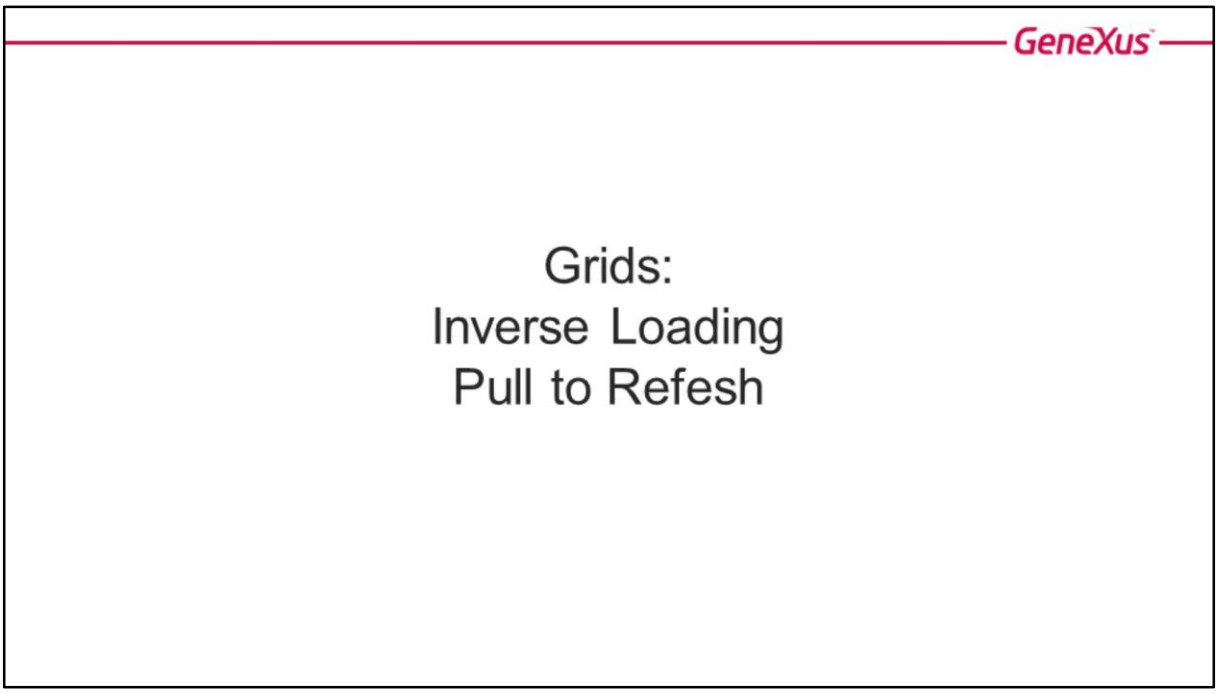
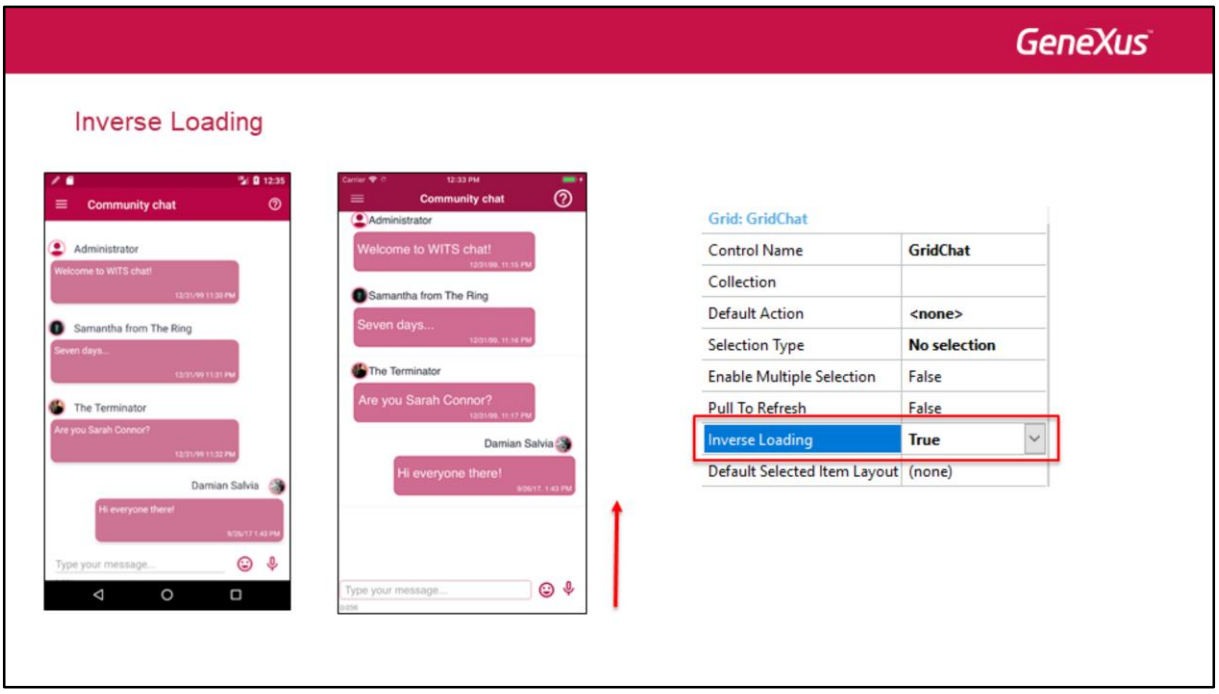
Smart Devices

GeneXus 15



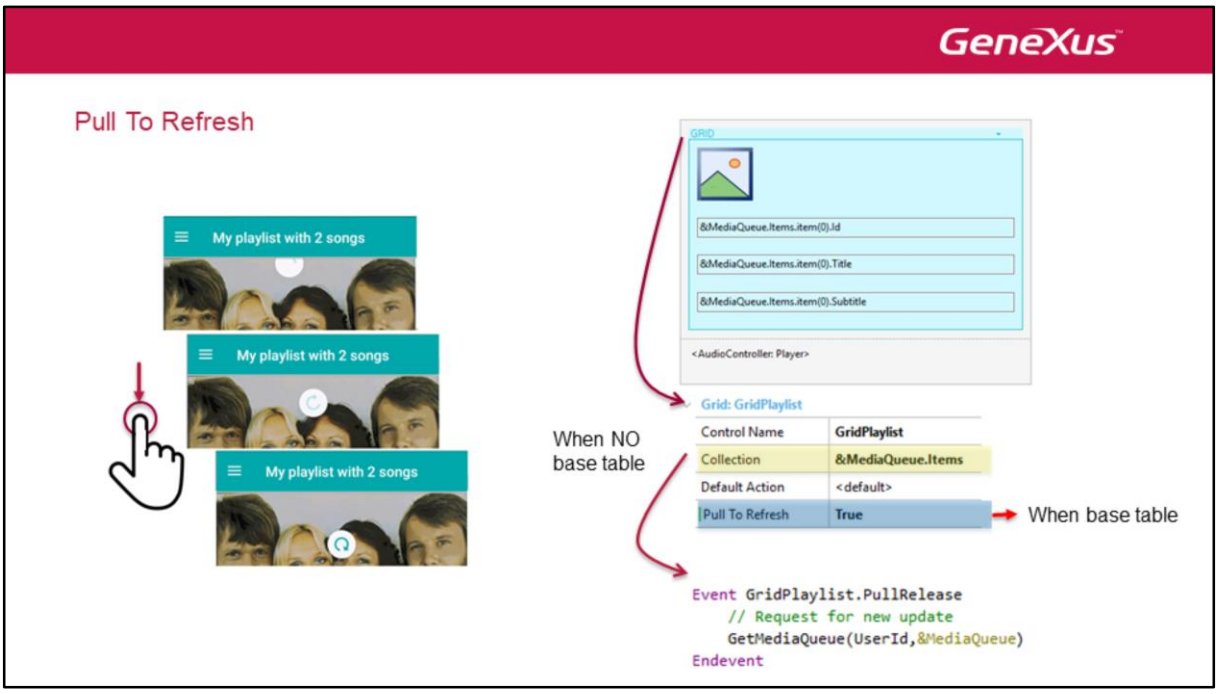
Veamos ahora un par de propiedades para los grids.



En el panel que implementa el chat de la aplicación vemos una nueva propiedad a nivel del grid, llamada Inverse Loading.
Por defecto está en False; la pasamos a True para implementar nuestro chat.

En el ejemplo, para simular el comportamiento de un chat simple, agregamos un order al grid por un atributo ChatTimestamp. De esta forma los registros más nuevos serán mostrados abajo del todo en el grid.

Aprovechemos y veamos la otra nueva propiedad, Pull To Refresh.



Todo usuario está familiarizado con esta feature. Si se tiene una pantalla scrollable que muestra datos que cambian (como por ejemplo las aplicaciones e Facebook o E-mail), el drag-down en la parte superior de la pantalla es una forma intuitiva de refrescarla para actualizar lo que el usuario está viendo.

Esto se consigue con la propiedad del Grid Pull To Refresh. Cuando está habilitada, el comportamiento default será preguntar al server si hay nueva información para devolver, en cuyo caso el grid de entrada será refrescado (y solo él, no el panel completo). Esto solo vale cuando el grid tiene table base. En otros casos (como el del ejemplo), podemos utilizar el nuevo evento **PullRelease**. Es disparado cuando el usuario final suelta la pantalla luego de hacer un pulling hacia abajo de ésta.

En el ejemplo simplemente se vuelve a cargar la variable colección &MediaQueue para que se refresquen las canciones que se muestran en el grid. Es decir, este grid estaba basado en un SDT &MediaQueue, que contenía todas las canciones, en una colección, correspondientes al usuario. Recordemos que ya habíamos visto este ejemplo cuando estudiamos el audiocontroller. Este era un data provider que iba a buscar, pasándole el usuario logueado a la aplicación, iba a buscar a una tabla de usuarios con las canciones adivinadas por el usuario, esas canciones y las iba a devolver en este formato SDT.

ControlValueChanging
new edit field event