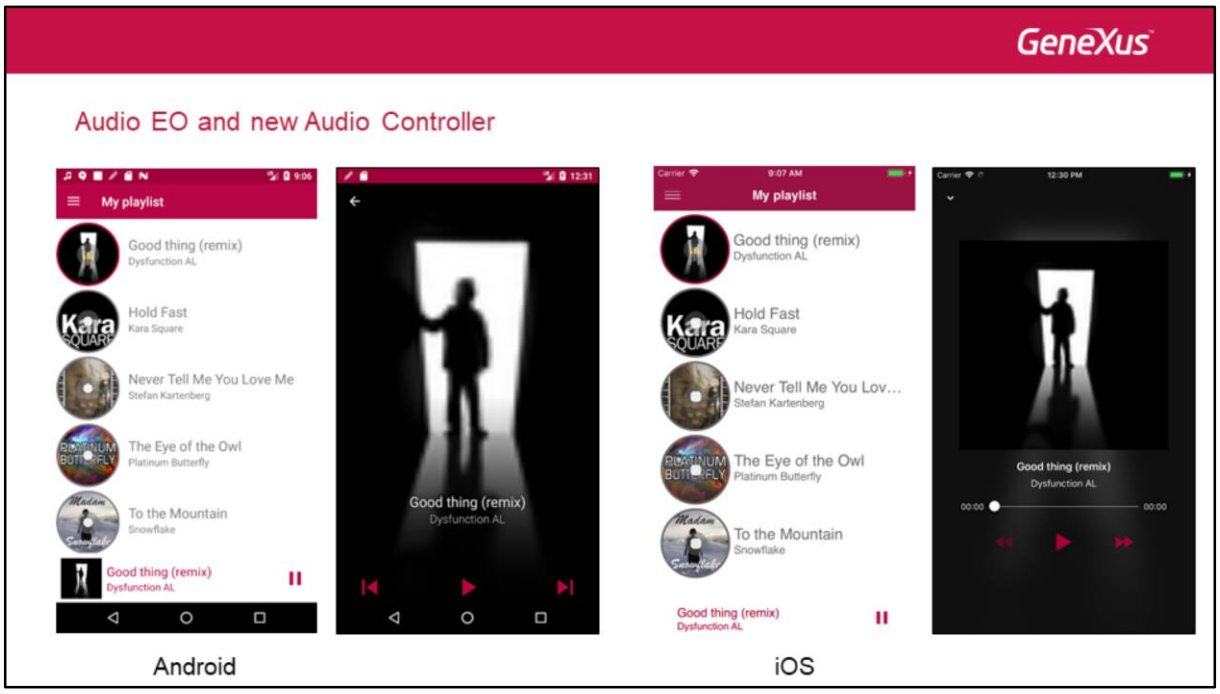
Smart Devices

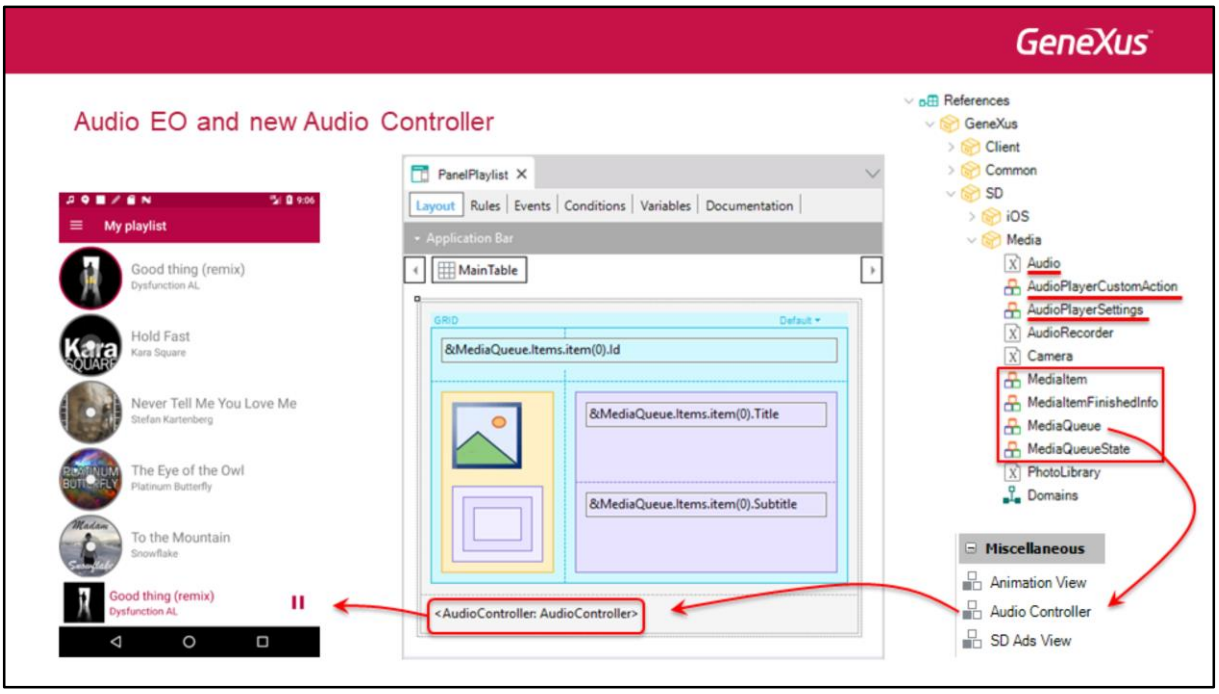
GeneXus 15

Audio



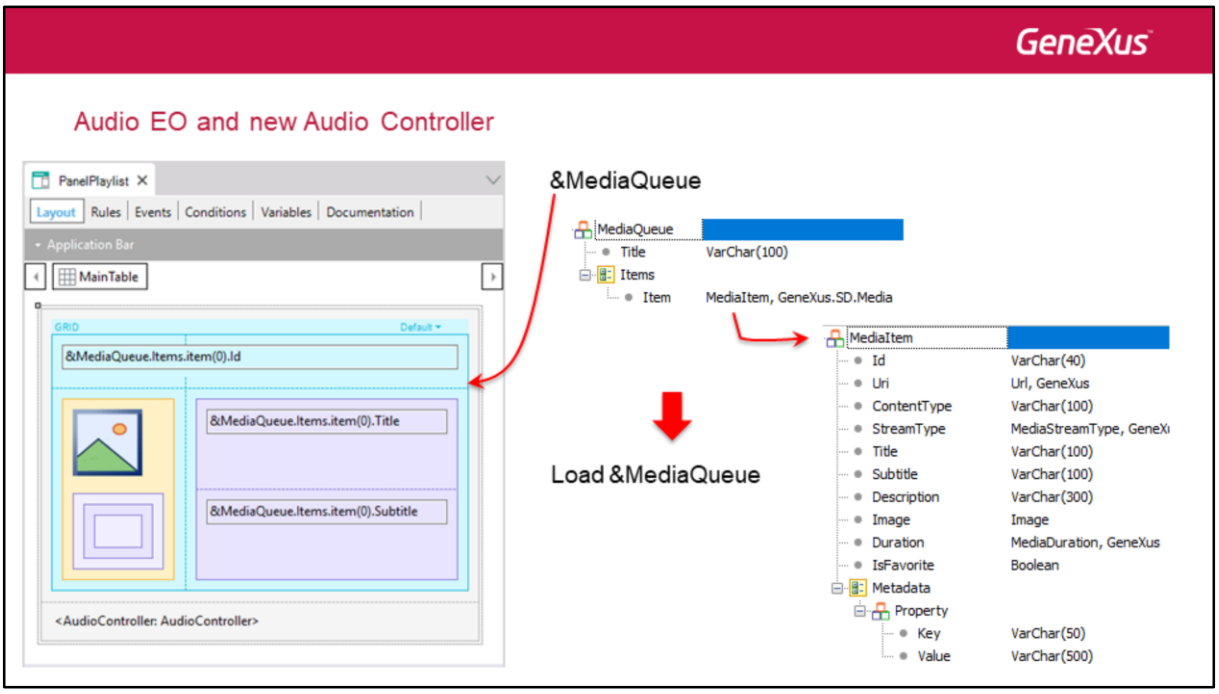
A partir de GeneXus 15 podemos manejar playlists e incorporar un audio player que puede mostrar la lista de reproducción en modo “mini”, -como estamos viendo aquí para Android, y aquí para iOS-, o en modo “fullScreen”, como podemos ver en las imágenes.

Aquí lo que estamos viendo, en las dos variedades, tanto para Android como para iOS es el panel Playlist, que es invocado desde el menú de la aplicación para ver mi playlist, la del usuario logueado, y que muestra primero en un grid todas las canciones que el usuario ha adivinado, y acá abajo tenemos un reproductor de audio que permite escucharlas.



El reproductor de playlists está implementado con el nuevo control AudioController. ¿Cómo sabe qué playlist reproducir, de dónde sacar los datos de esas canciones a ser reproducidas?

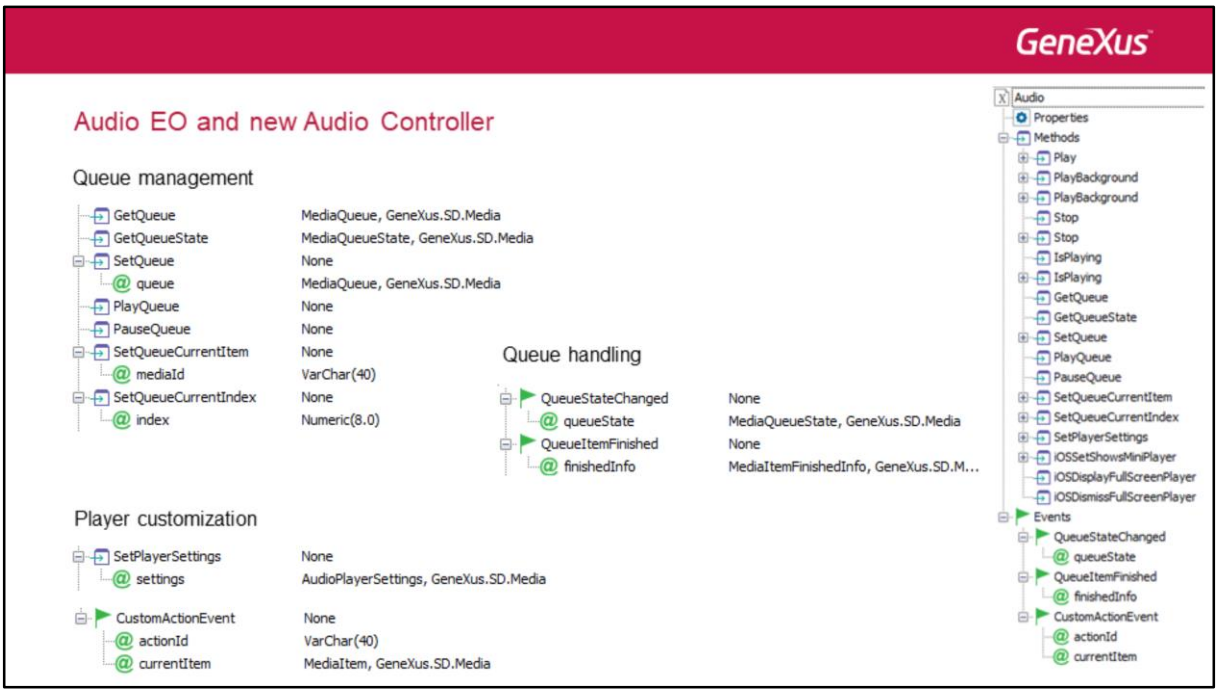
Si observamos el contenido del módulo Media, vemos que relacionados con la api Audio tenemos todos los SDTs que marcamos en esta imagen. Uno de ellos, MediaQueue es el que permite almacenar los datos necesarios para la cola de reproducción que será manejada por el AudioController.



Basta entonces con cargar una variable de tipo el SDT predefinido MediaQueue, que tiene una colección de ítems del tipo MediaItem, que almacena, a su vez, la info de una canción. En nuestro caso lo cargaremos de la base de datos, a partir de una transacción User que contendrá un subnivel Song con las canciones adivinadas por ese usuario.

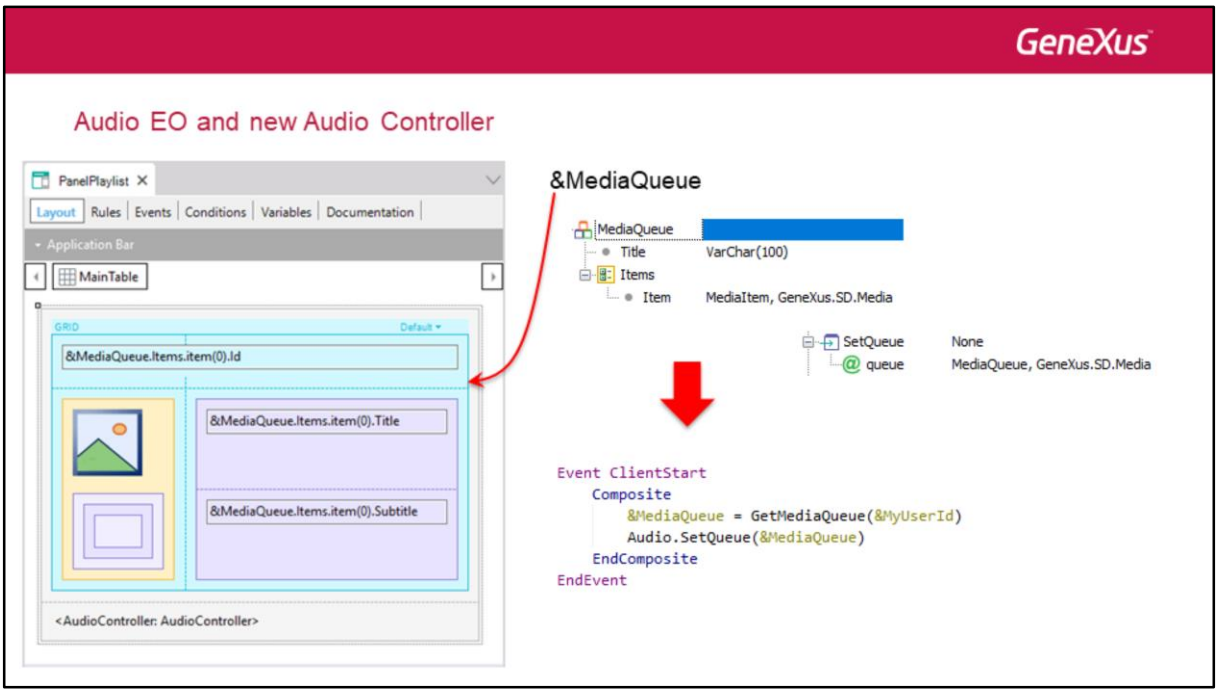
En el panel que vemos estamos utilizando la variable &MediaQueue tanto para mostrar su colección de ítems en un grid, como para reproducirlas con el audiocontroller.

Para poder manejar la reproducción, contamos con la api Audio.



A los métodos que ya tenía la api, vemos que agrega los que son para el manejo de colas de reproducción. Además agrega eventos.

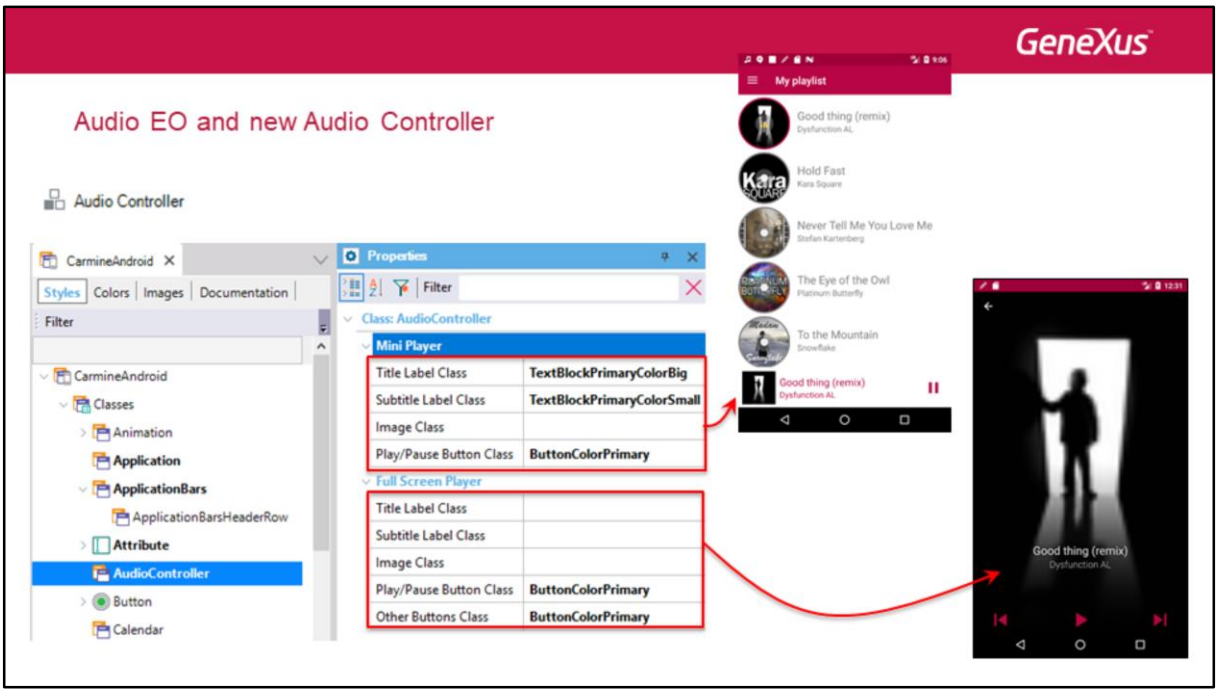
Aquí presentamos separados los que son para la administración de las colas, su manejo y los de personalización del reproductor (la posibilidad de agregar acciones distintas de las predefinidas (que son para detener: stop, pausar, continuar reproduciendo).



Volviendo a nuestro ejemplo, si solo vamos a dar la posibilidad de reproducir la cola con el reproductor, entonces alcanzará con programar el evento ClientStart como mostramos.

Aquí vemos que estamos utilizando un Data Provider, GetMediaQueue, pasándole el UserId. Este Data Provider lo que va a hacer es buscar en esta tabla de usuarios –va a filtrar por el usuario que recibe por parámetro- las canciones correspondientes a ese usuario. Ese Data Provider va a devolver un valor del tipo de este SDT, es decir, el título de la cola, y todos sus ítems (todas sus canciones). Y luego lo único que tenemos que hacer es invocar al método SetQueue del external object Audio, pasándole esa cola de reproducción.

Ahora la pregunta que se abre es: ¿cómo configuramos el diseño del reproductor?

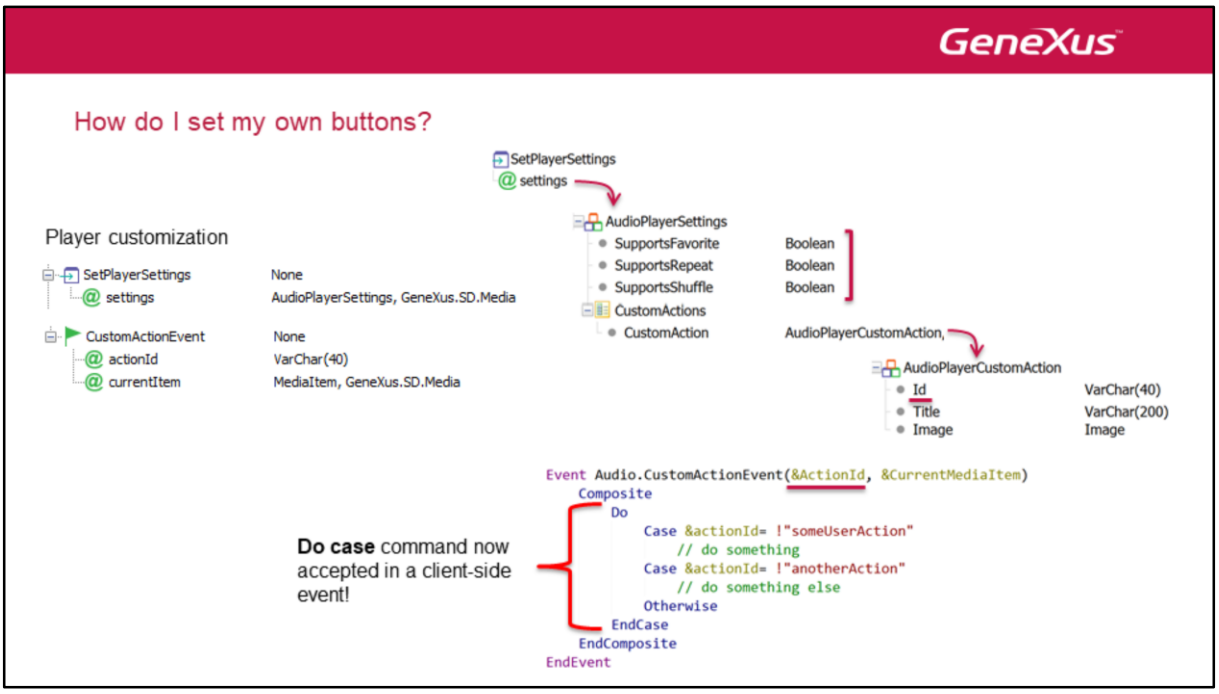


Bueno, el reproductor permite configurar en diseño las propiedades que vemos en la clase AudioController del theme.

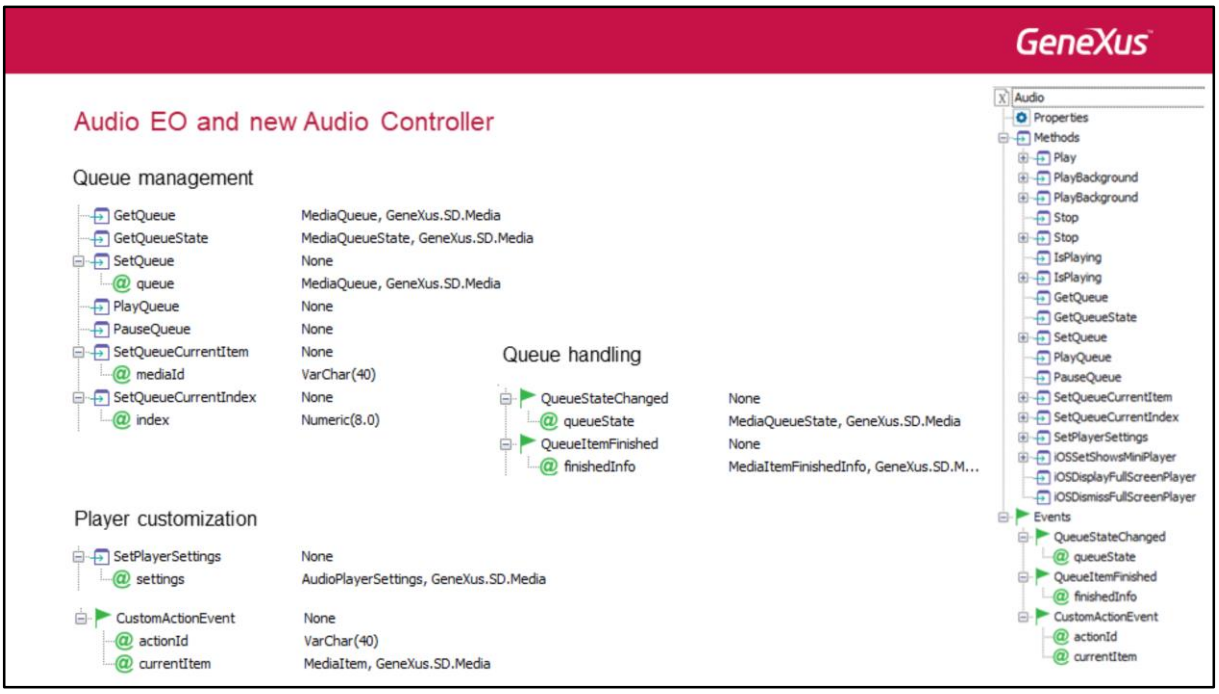
En nuestro ejemplo, vemos que en modo “mini” se ven en colores rosados la etiqueta del título y del subtítulo, con distintos tamaños de letra, y en modo “full” vemos en blanco título y subtítulo y los botones con los colores rosados de la aplicación.

Todo eso, entonces, se configuró dentro de la clase AudioController del grupo Mini Player y Full Screen Player.

Acá vemos algunos de los botones que corresponden a las acciones predefinidas del controlador de audio. Por ejemplo, la posibilidad de presionar Play o de pausar.



Pero podríamos querer definir nuestros propios botones, con acciones específicas para una cola de reproducción. Para ello tenemos los SDTs AudioPlayerSettings, que, como vemos, tiene un nivel fijo que permite definir si aparecen, si se utilizan o no alguna de las opciones predefinidas, por ejemplo favoritos, la opción de Repeat o de Shuffle. Y además vemos que tiene una colección de elementos CustomAction del tipo de este otro SDT, AudioPlayerCustomAction.
Aquí lo que se hace es definir el título, imagen y darle un identificador a la acción que va a corresponder a esos botones que vamos a agregar al controlador. La manera de identificarlos va a ser con este id de acá.
Entonces, luego a nivel de la api Audio, aparece esta opción SetPlayerSettings, que recibe como parámetro, veamos, una variable de este tipo. En definitiva sería como una metadata de los botones que se van a agregar junto con estas acciones de aquí, ¿verdad? Y luego necesitamos un evento que nos permita codificar qué hacer en el caso en que el usuario presione alguno de los botones personalizados estos de aquí. Para ello, tenemos el evento CustomActionEvent.
Codificaríamos el evento CustomActionEvent de la api Audio, pasándole, como vemos, estos dos parámetros. El ActionId va a ser el que corresponda, el que identifique al botón de este SDT (de la colección en definitiva), -va a corresponder al botón que el usuario presionó-. Entonces lo que hacemos es programa con un Do case –que ahora vemos que es aceptado en la version 15 de GeneXus en eventos del lado del cliente, antes recordemos que no-. Y bueno, lo que hacemos es, con ese do case, de acuerdo a cuál de estos botones identificado por el ActionId es el que disparó la acción, Bueno, hacer lo que corresponda en ese caso.



Vemos rápidamente que tenemos un montón de métodos y eventos para poder, por ejemplo, manejar la cola, QueueState Change, si cambió; o por ejemplo si terminó el item que se estaba reproduciendo actualmente; bueno, para setear la cola de reproducción, ya lo vimos; dar un play, pausar, etc.

Va a encontrar mucho más sobre esto en nuestro wiki.

Audio Recording