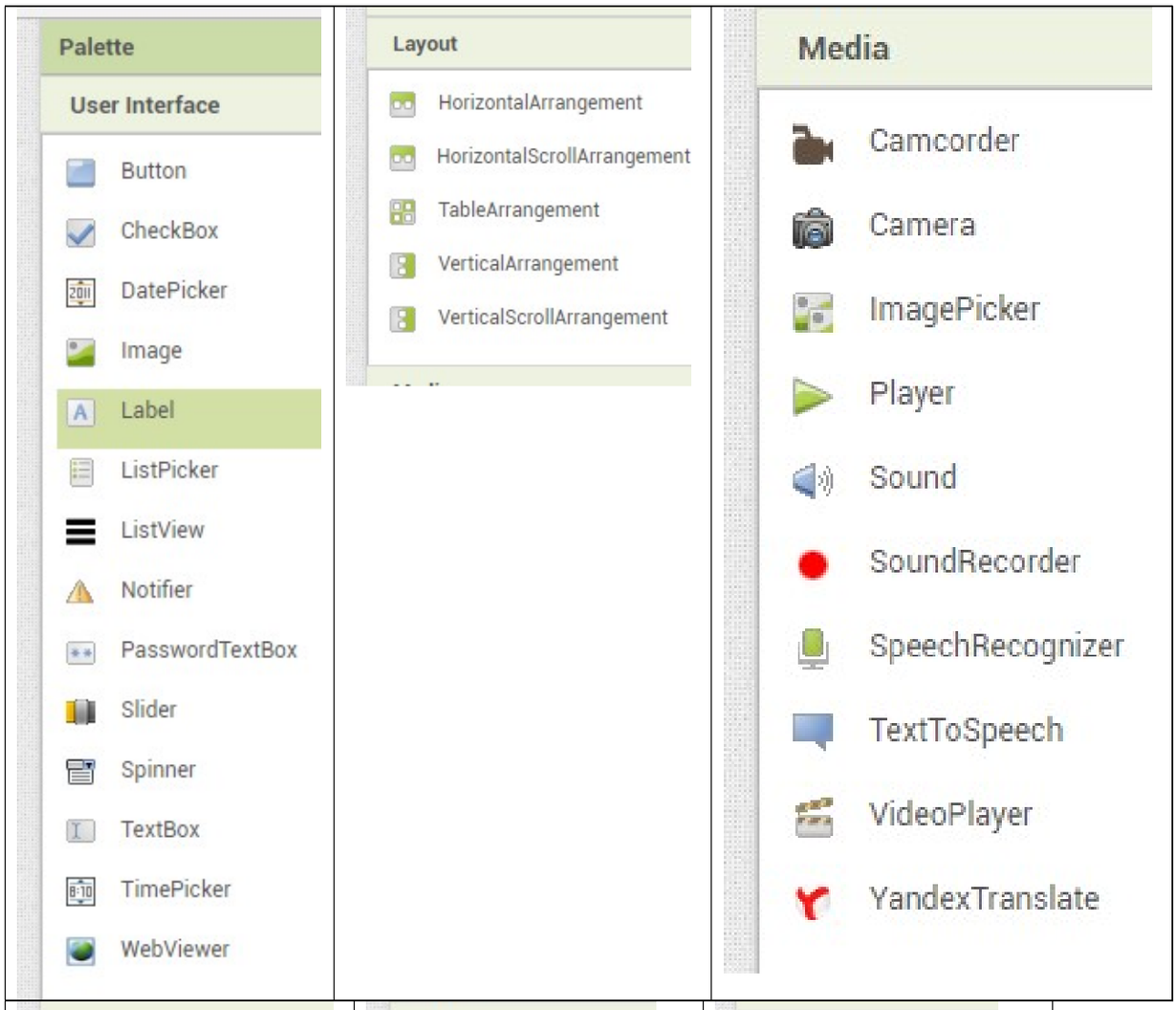
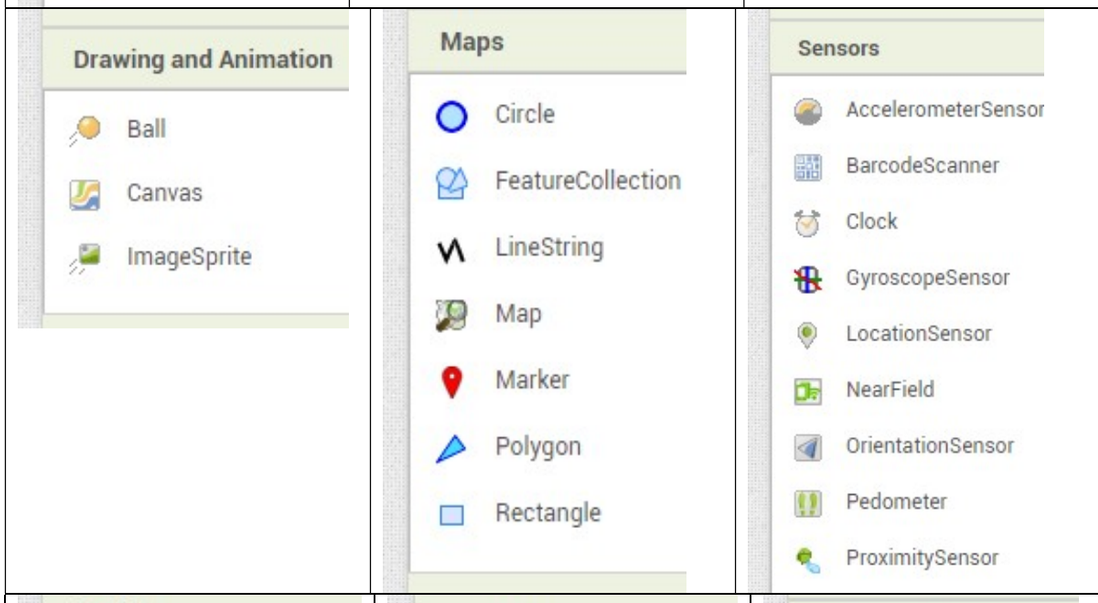
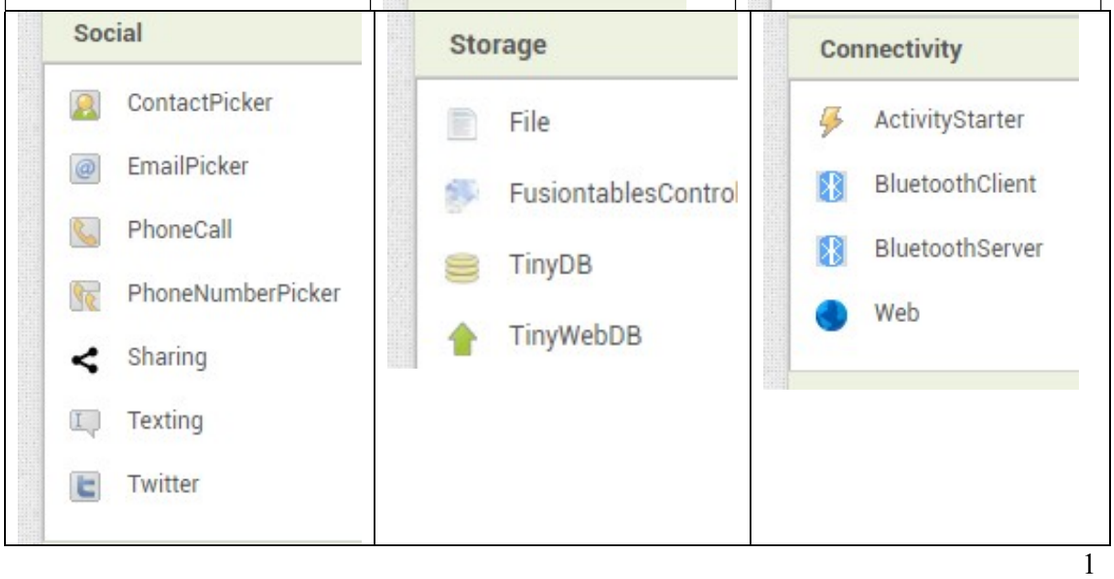

Palette
User Interface
Button
CheckBox
DatePicker
Image
Label
ListPicker
ListView
Notifier
PasswordTextBox
Slider
Spinner
TextBox
TimePicker
WebViewer
Layout
HorizontalArrangement
HorizontalScrollArrangement
TableArrangement
VerticalArrangement
VerticalScrollArrangement
Media
Camcorder
Camera
ImagePicker
Player
Sound
SoundRecorder
SpeechRecognizer
TextToSpeech
VideoPlayer
YandexTranslate


Drawing and Animation
Ball
Canvas
ImageSprite
Maps
Circle
FeatureCollection
LineString
Map
Marker
Polygon
Rectangle
Sensors
AccelerometerSensor
BarcodeScanner
Clock
GyroscopeSensor
LocationSensor
NearField
OrientationSensor
Pedometer
ProximitySensor


Social
ContactPicker
EmailPicker
PhoneCall
PhoneNumberPicker
Sharing
Texting
Twitter
Storage
File
FusiontablesControl
TinyDB
TinyWebDB
Connectivity
ActivityStarter
BluetoothClient
BluetoothServer
Web

Componentes de la interfaz de usuario: App Inventor para Android

Tabla de contenido

Botón

Botón con la capacidad de detectar clics. Se pueden cambiar muchos aspectos de su apariencia, así como si se puede hacer clic en él (`Enabled`), se puede cambiar en el Diseñador o en el Editor de bloques.

Propiedades

`BackgroundColor`
: Devuelve el color de fondo del botón

`Enabled`
: Si se establece, el usuario puede presionar la casilla de verificación para causar una acción.

`FontBold`
: Si está configurado, el texto del botón se muestra en negrita.

`FontItalic`
: Si se establece, el texto del botón se muestra en cursiva.

`FontSize`
: Tamaño del punto para el texto del botón.

`FontTypeface` (solo diseñador)
: Familia de fuentes para el texto del botón.

`Height`
: Altura del botón (tamaño y)

`Image`
: Imagen para mostrar en el botón.

`Shape` (solo diseñador)
: Especifica la forma del botón (predeterminado, redondeado, rectangular, oval). La forma no será visible si se muestra una imagen.

`ShowFeedback`
: Especifica si se debe mostrar un comentario visual para un botón que como una imagen como fondo.

`Text`
: Texto para mostrar en el botón.

`TextAlignment` (solo diseñador)
: Izquierda, centro o derecha.

`TextColor`
: Color para el texto del botón.

`Visible`
: Especifica si el componente debe ser visible en la pantalla. El valor es verdadero si el componente se muestra y falso si está oculto.

`Width`
: Ancho del botón (tamaño x).

Eventos

`Click()`
: El usuario tocó y soltó el botón.

`GotFocus()`
: Indica que el cursor se movió sobre el botón, por lo que ahora es posible hacer clic en él.

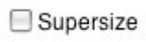
`LongClick()`
: El usuario mantuvo presionado el botón.

`LostFocus()`
: Indica que el cursor se alejó del botón, por lo que ya no es posible hacer clic en él.

`TouchDown()`
: Indica que el botón fue presionado hacia abajo.

`TouchUp()`
: Indica que un botón ha sido liberado.

Caja



Los componentes de la casilla de verificación pueden detectar los grifos de usuario y pueden cambiar su estado booleano en respuesta.

Un componente de casilla de verificación genera un evento cuando el usuario lo toca. Hay muchas propiedades que afectan a su apariencia que se pueden configurar en Designer o Blocks Editor.

Propiedades

`BackgroundColor`
: Color para el fondo de la casilla de verificación.

`Checked`
: Verdadero si la casilla está marcada, de lo contrario es falso.

`Enabled`
: Si se establece, el usuario puede presionar la casilla de verificación para causar una acción.

`Height`
: Compruebe la altura de la caja (tamaño y).

`Width`
: Ancho de la casilla de verificación (tamaño x).

`Text`
: Texto para mostrar en la casilla de verificación.

`TextColor`
: Color para el texto de la casilla de verificación.

`Visible`
: Si está establecido, la casilla de verificación está visible.

Eventos

`Click()`
: El usuario ha pulsado y ha liberado la casilla de verificación.

`GotFocus()`
: La casilla de verificación se convirtió en el componente enfocado.

`LostFocus()`
: La casilla de verificación dejó de ser el componente enfocado.

Selector de fechas

Un botón que, cuando se hace clic en él, abre un cuadro de diálogo emergente para permitirle al usuario seleccionar una fecha.

Propiedades

`BackgroundColor`
: Devuelve el color de fondo del botón

`Day`
: el día del mes que se eligió por última vez con DatePicker.

`Enabled`
: Si se establece, el usuario puede presionar la casilla de verificación para causar una acción.

`FontBold`
: Si está configurado, el texto del botón se muestra en negrita.

`FontItalic`
: Si se establece, el texto del botón se muestra en cursiva.

`FontSize`
: Tamaño del punto para el texto del botón.

`FontTypeface` (solo diseñador)
: Familia de fuentes para el texto del botón.

`Height`

`Image`
: Imagen para mostrar en el botón.

`Instant`
: Instante de fecha. Este instante se puede usar con el componente Reloj para la documentación de fechas, la conversión y la matriculación.

`Month`
: el número del mes que se eligió por última vez usando DatePicker. Tenga en cuenta que los meses comienzan en 1 = enero, 12 = diciembre.

`MonthInText`
: Devuelve el nombre del mes que se eligió por última vez utilizando DatePicker, en formato de texto.

`Shape` (solo diseñador)
: Especifica la forma del botón (predeterminado, redondeado, rectangular, oval). La forma no será visible si se muestra una imagen.

`ShowFeedback`
: Especifica si se debe mostrar un comentario visual para un botón que como una imagen como fondo.

`Text`
: Texto para mostrar en el botón.

`TextAlignment` (solo diseñador)
: Izquierda, centro o derecha.

`TextColor`
: Color para el texto del botón.

`Visible`
: Especifica si el componente debe ser visible en la pantalla. El valor es verdadero si el componente se muestra y falso si está oculto.

`Width`

`Year`
: el año que se eligió por última vez con DatePicker

Eventos

`AfterDateSet()`
: Evento que se ejecuta después de que el usuario elige una Fecha en el diálogo

`GotFocus()`
: Indica que el cursor se movió sobre el botón, por lo que ahora es posible hacer clic en él.

`LostFocus()`
: Indica que el cursor se alejó del botón, por lo que ya no es posible hacer clic en él.

`TouchDown()`
: Indica que el botón fue presionado hacia abajo.

`TouchUp()`
: Indica que un botón ha sido liberado.

Métodos

`LaunchPicker()`
: Inicia la ventana emergente DatePicker.

`SetDateToDisplay(number year, number month, number day)`
: Permite al usuario establecer la fecha que se mostrará cuando se abra el selector de fecha. Los valores válidos para el campo mes son 1-12 y 1-31 para el campo día.

`SetDateToDisplayFromInstant(InstantInTime instant)`
: Permite que el instante configure el año, mes y día que se mostrará cuando se abra el selector de fecha. Los instantes se utilizan en los componentes Clock, DatePicker y TimePicker.

Imagen

Componente para mostrar imágenes. La imagen para mostrar y otros aspectos de la apariencia de la imagen se pueden especificar en el Diseñador o en el Editor de bloques.

Propiedades

`Animation`
: Esta es una forma de animación limitada que puede asociar un pequeño número de tipos de movimiento a las imágenes. Los movimientos permitidos son ScrollRightSlow, ScrollRight, ScrollRightFast, ScrollLeftSlow, ScrollLeft, ScrollLeftFast y Stop

`Height`
`Picture`
`Visible`
: Especifica si el componente debe ser visible en la pantalla. El valor es verdadero si el componente se muestra y falso si está oculto.

`Width`

Eventos

ninguna

Métodos

ninguna

Etiqueta

Las etiquetas son componentes utilizados para mostrar texto.

Una etiqueta muestra texto especificado por la `Text` propiedad. Otras propiedades, todas las cuales pueden configurarse en Designer o Blocks Editor, controlan la apariencia y ubicación del texto.

Propiedades

`BackgroundColor`
: Color para el fondo de la etiqueta.

`FontBold`
: Si se establece, el texto de la etiqueta se muestra en negrita.

`FontItalic`
: Si se establece, el texto de la etiqueta se muestra en cursiva.

`FontSize`
: Tamaño del punto para el texto de la etiqueta.

`FontTypeface`
: Familia de fuentes para el texto de la etiqueta.

`HasMargins`
: Informa si la etiqueta aparece con márgenes o no. Los cuatro márgenes (izquierda, derecha, arriba, abajo) son los mismos. Esta propiedad no tiene ningún efecto en el diseñador, donde las etiquetas siempre se muestran con márgenes.

`HTMLFormat`
: Si es verdadero, establece el texto de la etiqueta en formato html; de lo contrario, es formato de texto sin formato. Nota: No todo el HTML es compatible.

`Height`
: Altura de la etiqueta (tamaño y)

`Width`
: Ancho de la etiqueta (tamaño x).

`Text`
: Texto para mostrar en la etiqueta.

`TextAlignment`
: Izquierda, centro o derecha.

`TextColor`
: Color para el texto de la etiqueta.

`Visible`
: Si se establece, la etiqueta es visible.

ListPicker

Un botón que, cuando se hace clic en, muestra una lista de textos para que el usuario pueda elegir. Los textos se pueden especificar a través del Diseñador o del Editor de bloques estableciendo la `ElementsFromString`propiedad en su concatenación separada por cadenas (por ejemplo, *opción 1, opción 2, opción 3*) o estableciendo la `Elements`propiedad en una lista en el editor de bloques.

Establecer la propiedad ShowFilterBar en true hará que la lista sea buscable. Otras propiedades afectan a la apariencia del botón (`TextAlignment`, `BackgroundColor`, etc.) y si se puede hacer clic en (`Enabled`).

Propiedades

`BackgroundColor`
: Devuelve el color de fondo del botón

`Elements`
: Lista de opciones para mostrar (como una lista)

`ElementsFromString`
: Lista de elecciones separadas por comas para usar

`Enabled`
: Si el ListPicker puede ser aprovechado

`FontBold` (solo diseñador)
: Si se establece, el texto del selector de lista se muestra en negrita.

`FontItalic` (solo diseñador)
: Si se establece, el texto del selector de lista se muestra en cursiva.

`FontSize` (solo diseñador)
: Tamaño del punto para el texto del selector de lista.

`FontTypeface` (solo diseñador)
: Familia de fuentes para el texto del selector de listas.

`Height`
: Altura de la caja (tamaño y)

`Image`
: Especifica la ruta de la imagen del botón. Si hay una Imagen y un Color de fondo, solo la Imagen estará visible.

`Selection`
: El elemento seleccionado. Cuando el programador lo cambia directamente, la propiedad SelectionIndex también se cambia al primer elemento en el ListPicker con el valor dado. Si el valor no aparece, SelectionIndex se establecerá en 0.

`SelectionIndex`
: El índice del elemento seleccionado actualmente, comenzando en 1. Si no se selecciona ningún elemento, el valor será 0. Si se intenta establecerlo en un número menor que 1 o mayor que el número de elementos en ListPicker, SelectionIndex se establecerá en 0, y la Selección se establecerá en el texto vacío.

`Shape` (solo diseñador)
: Especifica la forma del botón (predeterminado, redondeado, rectangular, oval). La forma no será visible si se muestra una imagen.

`ShowFeedback`
: Especifica si se debe mostrar un comentario visual para un botón que como una imagen como fondo.

`ShowFilterBar`
: Devuelve el estado actual de ShowFilterBar que indica si Search Filter Bar se mostrará en ListPicker o no

`Text`
: Título de texto para mostrar en selector de lista.

`TextAlignment` (solo diseñador)
: Izquierda, centro o derecha.

`TextColor`
: Color para texto

`Title`
: Título opcional que se muestra en la parte superior de la lista de opciones.

`Visible`
: Especifica si el componente debe ser visible en la pantalla. El valor es verdadero si el componente se muestra y falso si está oculto.

`Width`
: Ancho de caja (tamaño x).

`ItemTextColor`
: El color del texto de los elementos de ListPicker.

`ItemBackgroundColor`
: El color de fondo de los elementos ListPicker.

Eventos

`AfterPicking()`

Evento que se levantará después de que la actividad del selector devuelva su resultado y las propiedades se hayan completado.

`BeforePicking()`
Evento para subir cuando se hace clic en el botón del componente o la lista se muestra usando el bloque Abrir. Este evento ocurre antes de que se muestre la lista de elementos, y puede usarse para preparar la lista antes de que se muestre.

`GotFocus()`
Indica que el cursor se movió sobre el botón, por lo que ahora es posible hacer clic en él.

`LostFocus()`
Indica que el cursor se alejó del botón, por lo que ya no es posible hacer clic en él.

Métodos

`Open()`
Abre el selector, como si el usuario hiciera clic en él.

Vista de la lista

Este es un componente visible que permite colocar una lista de elementos de texto en su pantalla para mostrar.
La lista se puede establecer usando la propiedad ElementsFromString o usando el bloque Elementos en el editor de bloques.
Advertencia: este componente no funcionará correctamente en las pantallas que se pueden desplazar.

Propiedades

`BackgroundColor`
El color del fondo de la vista de lista.

`Elements`
Lista de elementos de texto para construir su lista.

`ElementsFromString`
Cree una lista con una serie de elementos de texto separados por comas como: queso, fruta, tocino, rábano. Cada palabra antes de la coma será un elemento en la lista.

`Height`
Determina la altura de la lista en la vista.

`Selection`
Devuelve el texto seleccionado por última vez en ListView.

`SelectionIndex`
El índice del elemento seleccionado actualmente, comenzando en 1. Si no se selecciona ningún elemento, el valor será 0. Si se intenta establecer esto en un número menor que 1 o mayor que el número de elementos en el ListView, SelectionIndex se establecerá en 0, y la Selección se establecerá en el texto vacío.

`ShowFilterBar`
Establece la visibilidad de ShowFilterBar. True mostrará la barra, False lo ocultará.

`TextColor`
El color del texto de los elementos de la vista de lista.

`Visible`
Especifica si el componente debe ser visible en la pantalla. El valor es verdadero si el componente se muestra y falso si está oculto.

`Width`
Determina el ancho de la lista en la vista.

Eventos

`AfterPicking()`
Evento simple que debe plantearse después de que se haya elegido un elemento en la lista. El elemento seleccionado está disponible en la propiedad Selección.

Métodos

ninguna

Notificador

El componente Notificador muestra cuadros de diálogo de alertas, mensajes y alertas temporales, y crea entradas de registro de Android a través de los siguientes métodos:

- ShowMessageDialog: muestra un mensaje que el usuario debe descartar presionando un botón.
- ShowChooseDialog: muestra un mensaje de dos botones para permitir al usuario elegir una de las dos respuestas, por ejemplo, sí o no, después de lo cual se genera el evento AfterChoosing.
- ShowTextDialog: permite al usuario ingresar texto en respuesta al mensaje, después del cual se genera el evento AfterTextInput.
- ShowAlert: muestra una alerta temporal que desaparece después de un corto tiempo.
- ShowProgressDialog: muestra una alerta con un spinner de carga que el usuario no puede descartar. Solo se puede descartar mediante el uso del bloque DismissProgressDialog.
- DismissProgressDialog: descarta el diálogo de progreso mostrado por ShowProgressDialog.
- LogError: registra un mensaje de error en el registro de Android.
- LogInfo: registra un mensaje de información en el registro de Android.
- LogWarning: registra un mensaje de advertencia en el registro de Android.
- Los mensajes en los cuadros de diálogo (pero no en la alerta) se pueden formatear utilizando las siguientes etiquetas HTML: <b>, <grande>, <blockquote>,
, <cite>, <dfn>, , , <small>, , <sub>, <sup>, <tt>. <u>
- También puede usar la etiqueta de fuente para especificar el color, por ejemplo, <font color = "blue">. Algunos de los nombres de color disponibles son aqua, negro, azul, fucsia, verde, gris, lima, granate, azul marino, oliva, violeta, rojo, plateado, azul verdoso, blanco y amarillo

Propiedades

`BackgroundColor`
: Especifica el color de fondo para las alertas (no para los cuadros de diálogo).

`NotifierLength` (solo diseñador)
: especifica el período de tiempo que se muestra la alerta, ya sea "corta" o "larga".

`TextColor`
: Especifica el color del texto para las alertas (no para los cuadros de diálogo).

Eventos

`AfterChoosing(text choice)`
: Evento después de que el usuario haya hecho una selección para ShowChooseDialog.

`AfterTextInput(text response)`
: Evento generado después de que el usuario haya respondido a ShowTextDialog.

Métodos

`DismissProgressDialog()`
: Descartar un cuadro de diálogo de progreso visualizado previamente

`LogError(text message)`
: Escribe un mensaje de error en el registro del sistema de Android. Consulte la documentación de Google Android para saber cómo acceder al registro.

`LogInfo(text message)`
: Escribe un mensaje de información en el registro de Android.

`LogWarning(text message)`
: Escribe un mensaje de advertencia en el registro de Android. Consulte la documentación de Google Android para saber cómo acceder al registro.

`ShowAlert(text notice)`
: Mostrar una notificación temporal

`ShowChooseDialog(text message, text title, text button1Text, text button2Text, boolean cancelable)`
: Muestra un cuadro de diálogo con dos botones, desde donde el usuario puede elegir. Si cancelable es cierto, habrá un botón CANCELAR adicional. Presionar un botón aumentará el evento AfterChoosing. El parámetro de "elección" para AfterChoosing será el texto en el botón que se presionó, o "Cancelar" si se presionó el botón CANCELAR.

`ShowMessageDialog(text message, text title, text buttonText)`
: Muestre un cuadro de diálogo de alerta con un solo botón que rechaza la alerta.

`ShowProgressDialog(text message, text title)`
: Muestra un cuadro de diálogo con un título y un mensaje opcionales (use cadenas vacías si no se desean). Este cuadro de diálogo contiene un artefacto giratorio para indicar que el programa está funcionando. No puede ser cancelado por el usuario, pero debe ser descartado por el programa App Inventor utilizando el bloque DismissProgressDialog.

`ShowTextDialog(text message, text title, boolean cancelable)`
: Muestra un cuadro de diálogo donde el usuario puede ingresar texto, después del cual se generará el evento AfterTextInput. Si cancelable es cierto, habrá un botón CANCELAR adicional. Al ingresar texto, se generará el evento AfterTextInput. El parámetro de

"respuesta" a AfterTextInput será el texto que se ingresó, o "Cancelar" si se presionó el botón CANCELAR.

PasswordTextBox

Los usuarios ingresan contraseñas en un componente de cuadro de texto de contraseña, que oculta el texto que se ha escrito en él.

Un cuadro de texto de contraseña es el mismo que el `TextBox` componente ordinario , excepto que no muestra los caracteres escritos por el usuario.
Puede obtener o establecer el valor del texto en el cuadro con la `Text` propiedad. Si `Text` está en blanco, puede usar la `Hint` propiedad para proporcionar al usuario una sugerencia sobre qué escribir. El `Hint` texto aparece como débil en la caja.

Los componentes del cuadro de texto de contraseña generalmente se usan con un componente de botón. El usuario toca el botón luego de ingresar el texto.

Métodos

`RequestFocus()`
: Establece PasswordTextBox activo.

Propiedades

`BackgroundColor`
: Color para el fondo del cuadro de texto.

`Enabled`
: Si se establece, el usuario puede ingresar una contraseña en el cuadro.

`FontBold`
: Si se establece, el texto se muestra en negrita.

`FontItalic`
: Si se establece, el texto se muestra en cursiva.

`FontSize`
: Tamaño del punto para el texto

`FontTypeface`
: Familia de fuentes para texto

`Height`
: Altura de la caja (tamaño y)

`Width`
: Ancho de caja (tamaño x).

`TextAlignment`
: Izquierda, centro o derecha.

`TextColor`
: Color para texto

`Hint`
: Pista de la contraseña.

`PasswordVisible`
: Visibilidad de la contraseña

Eventos

`GotFocus()`
: Box se convirtió en el componente enfocado.

`LostFocus()`
: Box ya no es el componente enfocado.

Pantalla

Componente de nivel superior que contiene todos los demás componentes en el programa

Propiedades

`AboutScreen`
: Información sobre la pantalla. Aparece cuando se selecciona "Acerca de esta aplicación" en el menú del sistema. Úselo para informar a los usuarios sobre su aplicación. En las aplicaciones de pantalla múltiple, cada pantalla tiene su propia información AboutScreen.

`AccentColor`
: Este es el color de acento utilizado para destacar y otros acentos de interfaz de usuario en las versiones más nuevas de Android. Los componentes afectados por esta propiedad incluyen los diálogos creados por el notificador, DateTimePicker y otros.

`ActionBar`
: De forma predeterminada, la barra de título coincidirá con las versiones anteriores de Android. Al habilitar esta opción, se reemplazará la barra de título predeterminada con una Barra de acción para que coincida con las versiones más recientes de Android. El ActionBar se puede diseñar utilizando las propiedades PrimaryColor y Theme. ActionBar solo se admite en temas distintos de "Clásico".

`AlignHorizontal`
: Un número que codifica cómo los contenidos de la pantalla se alinean horizontalmente. Las opciones son: 1 = alineado a la izquierda, 2 = centrado horizontalmente, 3 = alineado a la derecha.

`AlignVertical`
: Un número que codifica cómo el contenido de la disposición se alinea verticalmente. Las opciones son: 1 = alineado en la parte superior, 2 = centrado verticalmente, 3 = alineado en la parte inferior. La alineación vertical no tiene efecto si la pantalla es desplazable.

`BackgroundColor`

`AppName` (solo diseñador)
: Este es el nombre para mostrar de la aplicación instalada en el teléfono. Si AppName está en blanco, se establecerá con el nombre del proyecto cuando se construya el proyecto.

`BackgroundImage`
: La imagen de fondo de la pantalla.

`CloseScreenAnimation`
: La animación para cerrar la pantalla actual y volver a la pantalla anterior. Las opciones válidas son por defecto, fade, zoom, slidehorizontal, slidevertical y none

`Height`
: Altura de la pantalla (tamaño y)

`Icon` (solo diseñador)

`OpenScreenAnimation`
: La animación para cambiar a otra pantalla. Las opciones válidas son por defecto, fade, zoom, slidehorizontal, slidevertical y none

`PrimaryColor`
: Este es el color primario utilizado como parte del tema de Android, que incluye colorear la barra de título de la pantalla.

`PrimaryColorDark`
: Este es el color primario utilizado cuando se especifica que la propiedad Theme es oscura. Se aplica a una serie de elementos, incluida la barra de título de la pantalla.

`ScreenOrientation`
: La orientación de pantalla solicitada, especificada como un valor de texto. Los valores comúnmente utilizados son paisaje, retrato, sensor, usuario y no especificado. Consulte la documentación del desarrollador de Android para ActivityInfo.Screen_Orientation para obtener la lista completa de configuraciones posibles.

`Scrollable`
: Cuando se marca, habrá una barra de desplazamiento vertical en la pantalla, y la altura de la aplicación puede exceder la altura física del dispositivo. Cuando no está marcada, la altura de la aplicación está restringida a la altura del dispositivo.

`ShowListsAsJson` (solo diseñador)
: Si es falso, las listas se convertirán en cadenas usando la notación Lisp, es decir, como símbolos separados por espacios, por ejemplo, (a 1 b2 (cd). Si es verdadero, las listas aparecerán como en Json o Python, por ejemplo ["a", 1 , "b", 2, ["c", "d"]]. Esta propiedad solo aparece en la pantalla 1, y el valor de la pantalla 1 determina el comportamiento de todas las pantallas. La propiedad se establece de manera predeterminada en "falso", lo que significa que la aplicación El programador de Inventor debe establecerlo explícitamente en "verdadero" si se desea la sintaxis de JSON / Python. En algún momento en el futuro modificaremos el sistema para que los nuevos proyectos se creen con esta propiedad establecida en "verdadero" de manera predeterminada. ser impactado. El programador de App Inventor también puede establecerlo nuevamente en "falso" en proyectos más nuevos si así lo desea.

`ShowStatusBar`
: La barra de estado es la barra superior de la pantalla. Esta propiedad informa si la barra de estado está visible.

`Sizing` (solo diseñador)
: Si se establece en fijo, se crearán diseños de pantalla para una sola pantalla de tamaño fijo y se escalarán automáticamente. Si se configura como receptivo, los diseños de pantalla utilizarán la resolución real del dispositivo. Consulte la documentación sobre diseño receptivo en App Inventor para obtener más información. Esta propiedad solo aparece en Screen1 y controla el tamaño de todas las pantallas de la aplicación.

`Theme`

Selecciona el tema para la aplicación. El tema solo se puede configurar en tiempo de compilación y el acompañante se aproximará a los cambios durante el desarrollo en vivo. Las posibles opciones son Classic, que es lo mismo que las versiones anteriores de App Inventor; Dispositivo predeterminado, que da el mismo tema que la versión de Android que se ejecuta en el dispositivo y utiliza PrimaryColor para ActinBar y tiene botones claros; Texto de título negro, que es el tema predeterminado del dispositivo pero con texto de título negro, y oscuro, que es una versión oscura del tema predeterminado del dispositivo que utiliza PrimaryColorDark y tiene componentes de color gris oscuro.

TitleVisible
: La barra de título es la barra gris superior en la pantalla. Esta propiedad informa si la barra de título está visible.

Title
: El título del formulario, que aparece en la barra de título

TutorialURL
: Una URL que se abrirá en el panel lateral izquierdo (que se puede alternar una vez que esté abierta). Esto está destinado a proyectos que tienen un tutorial en línea como parte del proyecto. Por razones de seguridad, solo los tutoriales alojados en http://appinventor.mit.edu o vinculados desde nuestro acortador de URL (http://appinv.us) pueden usarse aquí. Otras URL serán ignoradas silenciosamente.

VersionCode (solo diseñador)
: Un valor entero que debe incrementarse cada vez que se crea un nuevo archivo de paquete de aplicación de Android (APK) para Google Play Store.

VersionName (solo diseñador)
: Una cadena que se puede cambiar para permitir que los usuarios de Google Play Store distingan entre las diferentes versiones de la aplicación.

Width
: Ancho de la pantalla (tamaño x)

Eventos

BackPressed()
: Botón de retroceso del dispositivo presionado.

ErrorOccurred(component component, text functionName, number errorNumber, text message)
: Evento generado cuando se produce un error. Solo algunos errores elevarán esta condición. Para esos errores, el sistema mostrará una notificación por defecto. Puede usar este controlador de eventos para prescribir un comportamiento de error diferente al predeterminado.

Initialize()
: Inicio de pantalla

OtherScreenClosed(text otherScreenName, any result)
: Evento levantado cuando se cierra otra pantalla y el control vuelve a esta pantalla.

ScreenOrientationChanged()
: La orientación de la pantalla cambió

Métodos

ninguna

Control deslizante

Un control deslizante es una barra de progreso que agrega un pulgar que se puede arrastrar. Puede tocar el dedo pulgar y arrastrar hacia la izquierda o hacia la derecha para establecer la posición del control deslizante del dedo pulgar. A medida que se arrastra el control deslizante del control deslizante, desencadenará el evento PositionChanged e informará la posición del control deslizante. La posición informada del control deslizante puede utilizarse para actualizar dinámicamente otro atributo de componente, como el tamaño de fuente de un cuadro de texto o el radio de una bola.

Propiedades

ColorLeft
: El color del control deslizante a la izquierda del pulgar.

ColorRight
: El color del control deslizante a la izquierda del pulgar.

MaxValue

Establece el valor máximo del control deslizante. Al cambiar el valor máximo también se restablece Thumbposition para que esté a medio camino entre el mínimo y el (nuevo) máximo. Si el nuevo máximo es menor que el mínimo actual, entonces el mínimo y el máximo se establecerán en este valor. Al establecer MaxValue se restablece la posición del pulgar a medio camino entre MinValue y MaxValue y se señala el evento PositionChanged.

`MinValue`
Establece el valor mínimo del control deslizante. Al cambiar el valor mínimo también se restablece Thumbposition para que esté a medio camino entre el mínimo (nuevo) y el máximo. Si el nuevo mínimo es mayor que el máximo actual, el mínimo y el máximo se establecerán en este valor. Al configurar MinValue se restablece la posición del pulgar a medio camino entre MinValue y MaxValue y se señala el evento PositionChanged.

`ThumbPosition`
Establece la posición del control deslizante pulgar. Si este valor es mayor que MaxValue, se establecerá en el mismo valor que MaxValue. Si este valor es menor que MinValue, se establecerá en el mismo valor que MinValue.

`ThumbEnabled`
Establece si se muestra o no el control deslizante.

`Visible`
Especifica si el componente debe ser visible en la pantalla. El valor es verdadero si el componente se muestra y falso si está oculto.

`Width`

Eventos

`PositionChanged(number thumbPosition)`
Indica que la posición del control deslizante ha cambiado.

Métodos

ninguna

Hilandero

Un componente giratorio que muestra una ventana emergente con una lista de elementos. Estos elementos se pueden establecer en Designer o Blocks Editor configurando la `ElementsFromString`propiedad en una concatenación separada por cadenas (por ejemplo, *opción 1, opción 2, opción 3*) o estableciendo la `Elements`propiedad en una lista en el editor de bloques. Los hilanderos se crean con el primer elemento ya seleccionado. Por lo tanto, seleccionarlo no genera un evento After Picking. Por lo tanto, es útil hacer que el primer elemento de Spinner no tenga opción, como "Seleccionar de abajo ...".

Propiedades

`Elements`
devuelve una lista de elementos de texto para elegir.

`ElementsFromString`
establece la lista de Spinner a los elementos pasados en la cadena separada por comas

`Height`
`Prompt`
Texto con el título actual para la ventana de Spinner

`Selection`
Devuelve el elemento seleccionado actual en la rueda giratoria

`SelectionIndex`
El índice del elemento seleccionado actualmente, comenzando en 1. Si no se selecciona ningún elemento, el valor será 0.

`Visible`
Especifica si el componente debe ser visible en la pantalla. El valor es verdadero si el componente se muestra y falso si está oculto.

`Width`

Eventos

`AfterSelecting(text selection)`
Evento llamado después de que el usuario selecciona un elemento de la lista desplegable.

Métodos

`DisplayDropdown()`

muestra la lista desplegable para la selección, la misma acción que cuando el usuario hace clic en la ruleta.

Caja de texto

Los usuarios ingresan texto en un componente de cuadro de texto.

El valor de texto inicial o ingresado por el usuario en un componente de cuadro de texto se encuentra en la `Text` propiedad. Si `Text` está en blanco, puede usar la `Hint` propiedad para proporcionar al usuario una sugerencia sobre qué escribir. El `Hint` texto aparece como débil en la caja.
La `MultiLine` propiedad determina si el texto puede tener más de una línea. Para un cuadro de texto de una sola línea, el teclado se cerrará automáticamente cuando el usuario presione la tecla Listo. Para cerrar el teclado de los cuadros de texto de líneas múltiples, la aplicación debe usar el método HideKeyboard o depender del usuario para presionar la tecla Atrás.
La `NumbersOnly` propiedad restringe el teclado para aceptar solo entradas numéricas.
Otras propiedades afectan a la apariencia del cuadro de texto (`TextAlignment` , `BackgroundColor` , etc.) y si se puede utilizar (`Enabled`).
Los cuadros de texto se usan generalmente con el `Button` componente, y el usuario hace clic en el botón cuando se completa la entrada de texto.
Si el texto ingresado por el usuario no se debe mostrar, utilícelo `PasswordTextBox` en su lugar.

Métodos

`HideKeyboard()`
: Ocultar el teclado Solo los cuadros de texto de líneas múltiples necesitan esto. Los cuadros de texto de una sola línea cierran el teclado cuando los usuarios presionan la tecla Listo.

`RequestFocus()`
: Establece el cuadro de texto activo.

Propiedades

`BackgroundColor`
: El color de fondo del cuadro de entrada. Puede elegir un color por nombre en el Diseñador o en el Editor de bloques. El color de fondo predeterminado es 'predeterminado' (aspecto tridimensional sombreado).

`Enabled`
: Si el usuario puede ingresar texto en este cuadro de entrada. Por defecto, esto es cierto.

`FontBold` (solo diseñador)
: Si la fuente para el texto debe ser negrita. Por defecto, no lo es.

`FontItalic` (solo diseñador)
: Si el texto debe aparecer en cursiva. Por defecto, no.

`FontSize`
: El tamaño de fuente para el texto. Por defecto, es 14.0 puntos.

`FontTypeface` (solo diseñador)
: La fuente para el texto. El valor se puede cambiar en el Diseñador.

`Height`
`Hint`
: Texto que debería aparecer débilmente en el cuadro de entrada para proporcionar una pista sobre lo que debe ingresar el usuario. Esto solo se puede ver si la `Text` propiedad está vacía.

`MultiLine`
: Si es verdadero, este cuadro de texto acepta múltiples líneas de entrada, que se ingresan usando la tecla de retorno. Para los cuadros de texto de una sola línea, hay una tecla Listo en lugar de una tecla de retorno, y al presionar Listo se oculta el teclado. La aplicación debe llamar al método HideKeyboard para ocultar el teclado de un cuadro de texto mutiline.

`NumbersOnly`
: Si es verdadero, este cuadro de texto solo acepta números como entrada de teclado. Los números pueden incluir un punto decimal y un signo negativo inicial opcional. Esto se aplica solo a la entrada del teclado. Incluso si NumbersOnly es verdadero, puede usar [set Text to] para ingresar cualquier texto.

`Text`
: El texto en el cuadro de entrada, que puede ser configurado por el programador en Designer o Blocks Editor, o puede ser ingresado por el usuario (a menos que la `Enabled` propiedad sea falsa).

`TextAlignment` (solo diseñador)
: Si el texto debe dejarse justificado, centrado o justificado a la derecha. Por defecto, el texto queda justificado.

`TextColor`

El color para el texto Puede elegir un color por nombre en el Diseñador o en el Editor de bloques. El color de texto predeterminado es negro.

`Visible`
Si el componente es visible

`Width`

Eventos

`GotFocus()`
Evento elevado cuando se selecciona este componente para la entrada, tal como por el usuario que lo toca.

`LostFocus()`
Evento generado cuando este componente ya no se selecciona para la entrada, como si el usuario tocara un cuadro de texto diferente.

TimePicker

Un botón que, cuando se hace clic en él, abre un cuadro de diálogo emergente para permitirle al usuario seleccionar una hora.

Propiedades

`BackgroundColor`
Devuelve el color de fondo del botón

`Enabled`
`FontBold` (solo diseñador)
`FontItalic` (solo diseñador)
`FontSize` (solo diseñador)
`FontTypeface` (solo diseñador)
`Height`
`Hour`
La hora de la última vez establecida usando el selector de tiempo. La hora es en un formato de 24 horas. Si la última vez configurada fue a las 11:53 p. M., Esta propiedad devolverá 23.

`Image`
Especifica la ruta de la imagen del botón. Si hay una Imagen y un Color de fondo, solo la Imagen estará visible.

`Instant`
Instante de tiempo. Este instante se puede usar con el componente Reloj para la documentación del tiempo, la conversión y la matriculación.

`Minute`
El minuto de la última vez configurada con el selector de tiempo

`Shape` (solo diseñador)
Especifica la forma del botón (predeterminado, redondeado, rectangular, oval). La forma no será visible si se muestra una imagen.

`ShowFeedback`
Especifica si se debe mostrar un comentario visual para un botón que como una imagen como fondo.

`Text`
`TextAlignment` (solo diseñador)
`TextColor`
`Visible`
Especifica si el componente debe ser visible en la pantalla. El valor es verdadero si el componente se muestra y falso si está oculto.

`Width`

Eventos

`AfterTimeSet()`
Este evento se ejecuta cuando un usuario ha establecido la hora en el cuadro de diálogo emergente.

`GotFocus()`
Indica que el cursor se movió sobre el botón, por lo que ahora es posible hacer clic en él.

`LostFocus()`
Indica que el cursor se alejó del botón, por lo que ya no es posible hacer clic en él.

Métodos

`LaunchPicker()`
: Inicia la ventana emergente de TimePicker.

`SetTimeToDisplay(number hour, number minute)`
: Permite al usuario establecer la hora que se mostrará cuando se abra el selector de tiempo. Los valores válidos para el campo de hora son 0-23 y 0-59 para el segundo campo.

`SetTimeToDisplayFromInstant(InstantInTime instant)`
: Permite que el instante establezca la hora y los minutos que se mostrarán cuando se abra el selector de tiempo. Los instantes se utilizan en los componentes Clock, DatePicker y TimePicker.

WebViewer

Componente para ver páginas web. La URL de inicio se puede especificar en el Diseñador o en el Editor de bloques. La vista se puede configurar para que siga los enlaces cuando se toca, y los usuarios pueden completar los formularios web. Advertencia: este no es un navegador completo. Por ejemplo, presionar la tecla de retroceso del hardware del teléfono saldrá de la aplicación, en lugar de retroceder en el historial del navegador.

Puede usar la propiedad WebViewer.WebViewString para comunicarse entre su aplicación y el código JavaScript que se ejecuta en la página Webviewer. En la aplicación, obtienes y configuras WebViewString. En WebViewer, incluye Javascript que hace referencia a la ventana. Objeto AppInventor, utilizando methoods y *setWebViewString (texto)* .
Por ejemplo, si el WebViewer se abre en una página que contiene el comando Javascript
document.write ("La respuesta es" + window.AppInventor.getWebViewString ());
y si configura WebView.WebVewString como "hola", se mostrará la página web
La respuesta es hola .
Y si la página web contiene Javascript que ejecuta el comando
windowAppInventor.setWebViewString ("hello from Javascript") ,
el valor de la propiedad WebViewString será
hola desde Javascript .

Propiedades

`CurrentPageTitle`
: Título de la página actualmente vista

`CurrentUrl`
: URL de la página actualmente vista. Esto podría ser diferente de la URL de inicio si las páginas nuevas fueron visitadas por los siguientes enlaces.

`FollowLinks`
: Determina si seguir enlaces cuando se tocan en el WebViewer. Si sigue los enlaces, puede usar GoBack y GoForward para navegar por el historial del navegador.

`Height`
`HomeUrl`
: URL de la página a la que inicialmente debe abrir WebViewer. Configurar esto cargará la página.

`IgnoreSslError`
: Determine si ignorar o no los errores de SSL. Establézcalo en verdadero para ignorar los errores. Úselo para aceptar certificados autofirmados de sitios web.

`PromptforPermission`
: Si es True, solicite al usuario de WebView que otorgue permiso para acceder a la API de geolocalización. Si es False, entonces suponga que se concede permiso.

`UsesLocation` (solo diseñador)
: Si se debe otorgar o no permiso a la aplicación para usar la API de geolocalización de Javascript. Esta propiedad solo está disponible en el diseñador.

`Visible`
: Especifica si el componente debe ser visible en la pantalla. El valor es verdadero si el componente se muestra y falso si está oculto.

`WebViewString`
: Obtiene la cadena de WebView, que se puede ver a través de Javascript en WebView como el objeto window.AppInventor

`Width`

Eventos

ninguna

Métodos

`boolean CanGoBack()`
: Devuelve verdadero si el WebViewer puede regresar a la lista de historial.

`boolean CanGoForward()`
: Devuelve verdadero si el WebViewer puede avanzar en la lista de historial.

`ClearCaches()`
: Borre los cachés de WebViewer

`ClearLocations()`
: Borre los permisos de ubicación almacenados.

`GoBack()`
: Regrese a la página anterior en la lista de historial. No hace nada si no hay una página anterior.

`GoForward()`
: Avanza a la siguiente página en la lista de historial. No hace nada si no hay una página siguiente.

`GoHome()`
: Carga la página de inicio URL. Esto sucede automáticamente cuando se cambia la URL de origen.

`GoToUrl(text url)`
: Cargue la página en la URL dada.

Componentes de diseño - App Inventor para Android

Tabla de contenido

- HorizontalArrangement
- HorizontalScrollArrangement
- TableArrangement
- VerticalArrangement
- VerticalScrollArrangement

HorizontalArrangement

Use un componente de disposición horizontal para visualizar un grupo de componentes dispuestos de izquierda a derecha.

Este componente es un elemento de formato en el que coloca los componentes que se deben mostrar de izquierda a derecha. Si desea que los componentes se muestren uno sobre otro, use `VerticalArrangement` en su lugar.

En un HorizontalArrangement, los componentes están dispuestos a lo largo del eje horizontal, alineados verticalmente en el centro.

Si la propiedad HeightArrangement's Height se establece en Automatic, la altura real de la disposición está determinada por el componente más alto en la disposición cuya propiedad Height no está configurada en Fill Parent. Si la propiedad Height de HorizontalArrangment está establecida en Automatic y solo contiene componentes cuyas propiedades de Height están configuradas como Fill Parent, la altura real de la disposición se calcula utilizando las alturas automáticas de los componentes. Si la propiedad HeightArrangement's Height está establecida en Automatic y está vacía, la altura será 100.

Si la propiedad Anchura horizontal de un arreglo se establece en Automático, el ancho real de la disposición se determina por la suma de los anchos de los componentes. **Si la propiedad Anchura horizontal de una disposición se establece en Automático, cualquier componente cuyas propiedades Ancho se configuren como Padre primario se comportará como si estuvieran configurados en Automático.**

Si la propiedad Ancho de HorizontalArrangement está configurada como Fill Parent o está especificada en píxeles, cualquier componente cuyas propiedades de ancho estén configuradas como Fill Parent ocupará igualmente el ancho no ocupado por otros componentes.

Propiedades

`AlignHorizontal`
: Un número que codifica cómo los contenidos de la disposición se alinean horizontalmente. Las opciones son: 1 = alineado a la izquierda, 2 = alineado a la derecha,

3 = centrado horizontalmente. La alineación no tiene efecto si el ancho de la disposición es automático.

`AlignVertical`
: Un número que codifica cómo el contenido de la disposición se alinea verticalmente. Las opciones son: 1 = alineado en la parte superior, 2 = alineado en la parte inferior, 3 = centrado verticalmente. La alineación no tiene efecto si la altura del arreglo es automática.

`BackgroundColor`
: Color de fondo para este componente

`Image`
: Imagen de fondo para este componente

`Visible`
: Si es verdadero, el componente y su contenido son visibles.

`Height`
: Altura de disposición horizontal (tamaño y).

`Width`
: Ancho de disposición horizontal (tamaño x).

HorizontalScrollArrangement

Un elemento de formato en el que colocar los componentes que se deben mostrar de izquierda a derecha. Si desea que los componentes se muestren uno sobre otro, use `VerticalArrangement`en su lugar.

Esta versión es desplazable.

Propiedades

`AlignHorizontal`
: Un número que codifica cómo los contenidos de la disposición se alinean horizontalmente. Las opciones son: 1 = alineado a la izquierda, 2 = alineado a la derecha, 3 = centrado horizontalmente. La alineación no tiene efecto si el ancho de la disposición es automático.

`AlignVertical`
: Un número que codifica cómo el contenido de la disposición se alinea verticalmente. Las opciones son: 1 = alineado en la parte superior, 2 = alineado en la parte inferior, 3 = centrado verticalmente. La alineación no tiene efecto si la altura del arreglo es automática.

`BackgroundColor`
: Devuelve el color de fondo del componente

`Height`
`HeightPercent`
`Image`
: Especifica la ruta de la imagen del componente. Si hay una Imagen y un Color de fondo, solo la Imagen estará visible.

`Visible`
: Especifica si el componente debe ser visible en la pantalla. El valor es verdadero si el componente se muestra y falso si está oculto.

`Width`
`WidthPercent`

TableArrangement

Utilice un componente de disposición de tabla para visualizar un grupo de componentes de forma tabular.

Este componente es un elemento de formato en el que coloca los componentes que se deben mostrar en forma de tabla.

En un TableArrangement, los componentes se organizan en una grilla de filas y columnas, con no más de un componente visible en cada celda. **Si varios componentes ocupan la misma celda, solo el último será visible.**

Dentro de cada fila, los componentes se alinean verticalmente en el centro.

El ancho de una columna está determinado por el componente más ancho en esa columna. Al calcular el ancho de la columna, el ancho automático se utiliza para los componentes cuya propiedad Ancho está configurada como Relleno principal. **Sin embargo, cada componente siempre llenará todo el ancho de la columna que ocupa.**

La altura de una fila está determinada por el componente más alto en esa fila cuya propiedad Height no está configurada como Fill Parent. Si una fila contiene solo componentes cuyas propiedades de Altura se configuran como Padre primario, la altura de la fila se calcula utilizando las alturas automáticas de los componentes.

Propiedades

`Visible`
: Si es verdadero, el componente y su contenido son visibles.

`Rows (number-of-rows)`
: El número de filas en la tabla.

`Columns (number-of-columns)`
: El número de columnas en la tabla.

`Height`
: Altura de disposición de la mesa (tamaño y)

`Width`
: Ancho de disposición de la tabla (tamaño x).

VerticalArrangement

Utilice un componente de disposición vertical para visualizar un grupo de componentes dispuestos de arriba a abajo, alineados a la izquierda.

Este componente es un elemento de formato en el que coloca los componentes que se deben mostrar uno debajo del otro. El primer componente secundario se almacena en la parte superior, el segundo debajo de él, y así sucesivamente. Si desea que los componentes se muestren uno al lado del otro, use `HorizontalArrangement` en su lugar.

En VerticalArrangement, los componentes están dispuestos a lo largo del eje vertical, alineados a la izquierda.

Si la propiedad Ancho de una disposición vertical está establecida en Automático, el ancho real de la disposición está determinado por el componente más ancho en la disposición cuya propiedad de Ancho no está configurada como Relleno principal. Si la propiedad Ancho de una disposición vertical está establecida en Automático y contiene solo componentes cuyas propiedades de Ancho están configuradas como Primaria de relleno, el ancho real de la disposición se calcula utilizando los anchos automáticos de los componentes. Si la propiedad Width de VerticalArrangement está establecida en Automatic y está vacía, el ancho será 100.

Si la propiedad Height de VerticalArrangement se establece en Automatic, la altura real de la disposición se determina por la suma de las alturas de los componentes. **Si la propiedad Height de VerticalArrangement se establece en Automatic, cualquier componente cuyas propiedades de Height se establezcan en Fill Parent se comportará como si estuvieran configuradas en Automatic.**
Si una propiedad VerticalArrangement's Height está configurada como Fill Parent o está especificada en píxeles, cualquier componente cuyas propiedades Height se configuren como Fill Parent también ocupará la altura no ocupada por otros componentes.

Propiedades

`AlignHorizontal`
: Un número que codifica cómo los contenidos de la disposición se alinean horizontalmente. Las opciones son: 1 = alineado a la izquierda, 2 = centrado horizontalmente, 3 = alineado a la derecha. La alineación no tiene efecto si el ancho de la disposición es automático.

`AlignVertical`
: Un número que codifica cómo el contenido de la disposición se alinea verticalmente. Las opciones son: 1 = alineado en la parte superior, 2 = centrado verticalmente, 3 = alineado en la parte inferior. La alineación no tiene efecto si la altura del arreglo es automática.

`BackgroundColor`
: Color de fondo para este componente

`Image`
: Imagen de fondo para este componente

`Visible`
: Si es verdadero, el componente y su contenido son visibles.

`Height`
: Altura de disposición vertical (tamaño y).

`Width`
: Ancho de disposición vertical (tamaño x).

VerticalScrollArrangement

Un elemento de formato en el que colocar los componentes que se deben mostrar uno debajo del otro. (El primer componente secundario se almacena en la parte superior, el segundo debajo, etc.) Si desea que los componentes se muestren uno al lado del otro, use `HorizontalArrangement` en su lugar.

Esta versión es desplazable

Propiedades

`AlignHorizontal`
: Un número que codifica cómo los contenidos de la disposición se alinean horizontalmente. Las opciones son: 1 = alineado a la izquierda, 2 = alineado a la derecha, 3 = centrado horizontalmente. La alineación no tiene efecto si el ancho de la disposición es automático.

`AlignVertical`
: Un número que codifica cómo el contenido de la disposición se alinea verticalmente. Las opciones son: 1 = alineado en la parte superior, 2 = alineado en la parte inferior, 3 = centrado verticalmente. La alineación no tiene efecto si la altura del arreglo es automática.

`BackgroundColor`
: Devuelve el color de fondo del componente

`Height`
`HeightPercent`
`Image`
: Especifica la ruta de la imagen del componente. Si hay una Imagen y un Color de fondo, solo la Imagen estará visible.

`Visible`
: Especifica si el componente debe ser visible en la pantalla. El valor es verdadero si el componente se muestra y falso si está oculto.

`Width`
`WidthPercent`

Componentes de medios - App Inventor para Android

Tabla de contenido

Videocámara

Un componente para grabar un video usando la videocámara del dispositivo. Después de grabar el video, el nombre del archivo en el teléfono que contiene el clip está disponible como argumento para el evento AfterRecording. El nombre del archivo se puede usar, por ejemplo, para establecer la propiedad fuente de un componente VideoPlayer.

Propiedades

ninguna

Eventos

`AfterRecording(text clip)`
: Indica que se grabó un video con la cámara y proporciona la ruta a la imagen almacenada.

Métodos

`RecordVideo()`
Graba un video y luego plantea el evento AfterRecoding.

Cámara

Use un componente de cámara para tomar una foto en el teléfono.

La cámara es un componente no visible que toma una fotografía con la cámara del dispositivo. Después de tomar la fotografía, la ruta al archivo en el teléfono que contiene la imagen está disponible como argumento para el evento AfterPicture. La ruta se puede usar, por ejemplo, como la propiedad Imagen de un componente de imagen.

Propiedades

ninguna

Métodos

`TakePicture()`
Abre la cámara del teléfono para permitir que se tome una fotografía.

Eventos

`AfterPicture(Text image)`
Llamado después de tomar la foto. El argumento de texto `image` es la ruta que se puede usar para ubicar la imagen en el teléfono.

ImagePicker

Un botón de propósito especial. Cuando el usuario toca un selector de imágenes, aparece la galería de imágenes del dispositivo y el usuario puede elegir una imagen. Después de seleccionar una imagen, se guarda y la `Selected`propiedad será el nombre del archivo donde se almacena la imagen. Para no llenar el almacenamiento, se almacenará un máximo de 10 imágenes. Al seleccionar más imágenes, se eliminarán las imágenes anteriores, en orden de mayor a mayor.

Propiedades

`BackgroundColor`
Devuelve el color de fondo del botón

`Enabled`
`FontBold` (solo diseñador)
`FontItalic` (solo diseñador)
`FontSize` (solo diseñador)
`FontTypeface` (solo diseñador)
`Height`
`Image`
Especifica la ruta de la imagen del botón. Si hay una Imagen y un Color de fondo, solo la Imagen estará visible.

`Selection`
Ruta al archivo que contiene la imagen que se seleccionó.

`Shape` (solo diseñador)
Especifica la forma del botón (predeterminado, redondeado, rectangular, oval). La forma no será visible si se muestra una imagen.

`ShowFeedback`
Especifica si se debe mostrar un comentario visual para un botón que como una imagen como fondo.

`Text`
`TextAlignment` (solo diseñador)
`TextColor`
`Visible`
Especifica si el componente debe ser visible en la pantalla. El valor es verdadero si el componente se muestra y falso si está oculto.

`Width`

Eventos

`AfterPicking()`

El evento simple que se generará después de la actividad del selector devuelve su resultado y las propiedades se han completado.

`BeforePicking()`
Evento simple para plantear cuando se hace clic en el componente pero antes de que se inicie la actividad del selector.

`GotFocus()`
Indica que el cursor se movió sobre el botón, por lo que ahora es posible hacer clic en él.

`LostFocus()`
Indica que el cursor se alejó del botón, por lo que ya no es posible hacer clic en él.

Métodos

`Open()`
Abre el selector, como si el usuario hiciera clic en él.

Jugador

Componente multimedia que reproduce audio y controla la vibración del teléfono. El nombre de un campo multimedia se especifica en la `Source`propiedad, que se puede establecer en el Diseñador o en el Editor de bloques. El tiempo de una vibración se especifica en el Editor de bloques en milisegundos (milésimas de segundo).
Para formatos de audio admitidos, vea Formatos de medios compatibles con Android .
Este componente es mejor para archivos de sonido largos, como canciones, mientras que el `Sound`componente es más eficiente para archivos cortos, como efectos de sonido.

Propiedades

`IsPlaying`
Informa si los medios están reproduciendo

`Loop`
Si es verdadero, el jugador dará un bucle cuando se reproduce. Establecer Loop mientras el reproductor está jugando afectará la reproducción actual.

`PlayOnlyInForeground`
Si es verdadero, el reproductor pausará la reproducción cuando salga de la pantalla actual; si es falso (opción predeterminada), el reproductor continúa reproduciéndose siempre que se muestre o no la pantalla actual.

`Source`
`Volume`
Establece el volumen a un número entre 0 y 100

Eventos

`Completed()`
Indica que los medios han llegado al final

`OtherPlayerStarted()`
Este evento se señala cuando otro jugador ha comenzado (y el jugador actual está jugando o en pausa, pero no se detuvo).

Métodos

`Pause()`
Suspende la reproducción de los medios si se está reproduciendo.

`Start()`
Reproduce los medios. Si se pausó previamente, se reanuda la reproducción. Si se detuvo anteriormente, comienza desde el principio.

`Stop()`
Detiene la reproducción de los medios y busca el comienzo de la canción.

`Vibrate(number milliseconds)`
Vibra por un número específico de milisegundos.

Sonar

Un componente multimedia que reproduce archivos de sonido y opcionalmente vibra durante el número de milisegundos (milésimas de segundo) especificado en el Editor de bloques. El nombre del archivo de sonido para reproducir se puede especificar en el Diseñador o en el Editor de bloques.

Para conocer los formatos de archivos de sonido compatibles, consulte Formatos de medios compatibles con Android .

Este `Sound`componente es mejor para archivos de sonido cortos, como efectos de sonido, mientras que el `Player`componente es más eficiente para sonidos más largos, como canciones.

Propiedades

`MinimumInterval`
: El intervalo mínimo, en milisegundos, entre sonidos. Si reproduce un sonido, todas las llamadas Play () adicionales se ignorarán hasta que haya transcurrido el intervalo.

`Source`
: El nombre del archivo de sonido. Solo ciertos formatos son compatibles. Ver http://developer.android.com/guide/appendix/media-formats.html.

Eventos

ninguna

Métodos

`Pause()`
: Pausa para reproducir el sonido si se está reproduciendo.

`Play()`
: Reproduce el sonido.

`Resume()`
: Continúa reproduciendo el sonido después de una pausa.

`Stop()`
: Detiene la reproducción del sonido si se está reproduciendo.

`Vibrate(number millisecs)`
: Vibra por la cantidad especificada de milisegundos.

Grabador de sonido

Componente multimedia que graba audio

Propiedades

`SavedRecording`
: Especifica la ruta al archivo donde se debe almacenar la grabación. Si esta propiedad es la cadena vacía, iniciar una grabación creará un archivo en una ubicación adecuada. Si la propiedad no es la cadena vacía, debe especificar una ruta completa a un archivo en un directorio existente, incluido un nombre de archivo con la extensión .3gp.

Eventos

`AfterSoundRecorded(text sound)`
: Proporciona la ubicación del sonido recién creado.

`StartedRecording()`
: Indica que la grabadora se ha iniciado y puede detenerse.

`StoppedRecording()`
: Indica que la grabadora se ha detenido y puede reiniciarse.

Métodos

`Start ()`
: Comienza la grabación.

`Stop ()`
: Detiene la grabación.

SpeechRecognizer

Utilice un componente de reconocimiento de voz para escuchar al usuario hablar y convertir el sonido hablado en texto utilizando la función de reconocimiento de voz de Android.

Propiedades

`Result`
: El último texto producido por el reconocedor.

Métodos

`GetText()`
: Pide al usuario que hable y convierte el discurso en texto. Señala el `AfterGettingText` evento cuando el resultado está disponible.

Eventos

`AfterGetting(Text result)`
: Señalizado después de que el reconocedor haya producido el texto. El argumento es el resultado del texto que se produjo.

`BeforeGettingText()`
: Señalizado justo antes de que se llame al reconocedor.

Texto a voz

El componente TestToSpeech habla un texto dado en voz alta. Puede establecer el tono y la velocidad de la palabra.

También puede establecer un idioma suministrando un código de idioma. Esto cambia la pronunciación de las palabras, no el idioma real hablado. Por ejemplo, configurar el idioma al francés y hablar en inglés parecerá que alguien habla inglés (en) con acento francés.

También puede especificar un país suministrando un código de país. Esto puede afectar la pronunciación. Por ejemplo, el inglés británico (GBR) sonará diferente al inglés de EE. UU. (EE. UU.). No todos los códigos de país afectarán a todos los idiomas.

Los idiomas y países disponibles dependen del dispositivo en particular, y se pueden enumerar con las propiedades AvailableLanguages y AvailableCountries.

Propiedades

`AvailableCountries`
: Lista de los códigos de país disponibles en este dispositivo para usar con TextToSpeech. Consulte la documentación del desarrollador de Android en los idiomas admitidos para encontrar el significado de estas abreviaturas.

`AvailableLanguages`
: Lista de los idiomas disponibles en este dispositivo para usar con TextToSpeech. Consulte la documentación del desarrollador de Android en los idiomas admitidos para encontrar el significado de estas abreviaturas.

`Country`
: Código de país para usar en la generación de habla. Esto puede afectar la pronunciación. Por ejemplo, el inglés británico (GBR) sonará diferente al inglés de EE. UU. (EE. UU.). No todos los códigos de país afectarán a todos los idiomas.

`Language`
: Establece el idioma para TextToSpeech. Esto cambia la forma en que se pronuncian las palabras, no el idioma real que se habla. Por ejemplo, establecer el idioma en francés y hablar en inglés sonará como si alguien hablara inglés con acento francés.

`Pitch`
: Establece el tono para TextToSpeech. Los valores deben estar entre 0 y 2, donde los valores más bajos disminuyen el tono de la voz sintetizada y aumentan los valores mayores.

`Result`
`SpeechRate`
: Establece el SpeechRate para TextToSpeech. Los valores deben estar entre 0 y 2, donde los valores más bajos ralentizan el tono y los valores más altos lo aceleran.

Eventos

`AfterSpeaking(boolean result)`
: Evento que se levantará después de que se hable el mensaje.

`BeforeSpeaking()`
: Evento que se levanta cuando se invoca Speak, antes de que se hable el mensaje.

Métodos

`Speak(text message)`
: Habla el mensaje dado.

Reproductor de video

Un componente multimedia capaz de reproducir videos. Cuando se ejecuta la aplicación, VideoPlayer se mostrará como un rectángulo en la pantalla. Si el usuario toca el rectángulo, los controles aparecerán para reproducirse / pausar, saltar hacia adelante y saltar hacia atrás dentro del video. La aplicación también puede controlar el comportamiento llamando a los `Start`, `Pause`y `SeekTo`métodos.
Los archivos de video deben estar en formatos 3GPP (.3gp) o MPEG-4 (.mp4). Para obtener más información sobre formatos legales, consulte Formatos de medios compatibles con Android .

App Inventor para Android solo permite archivos de video de menos de 1 MB y limita el tamaño total de una aplicación a 5 MB, no todos están disponibles para archivos multimedia (video, audio y sonido). Si sus archivos multimedia son demasiado grandes, es posible que obtenga errores al empacar o instalar su aplicación, en cuyo caso debe reducir la cantidad de archivos multimedia o sus tamaños. La mayoría del software de edición de video, como Windows Movie Maker y Apple iMovie, puede ayudarlo a reducir el tamaño de los videos acortándolos o recodificando el video en un formato más compacto.

También puede establecer la fuente de medios en una URL que apunta a una transmisión de video, pero la URL debe apuntar al archivo de video en sí, no a un programa que reproduzca el video.

Propiedades

`FullScreen`
`Height`
`Source`
: El "camino" al video. Por lo general, este será el nombre del archivo de video, que debe agregarse en el Diseñador.

`Visible`
: Especifica si el componente debe ser visible en la pantalla. El valor es verdadero si el componente se muestra y falso si está oculto.

`Volume`
: Establece el volumen en un número entre 0 y 100. Los valores menores que 0 se tratarán como 0 y los valores superiores a 100 se tratarán como 100.

`Width`

Eventos

`Completed()`
: Indica que el video ha llegado al final

Métodos

`number GetDuration()`
: Devuelve la duración del video en milisegundos.

`Pause()`
: Pausa la reproducción del video. La reproducción se puede reanudar en el mismo lugar llamando al `Start`método.

`SeekTo(number ms)`
: Busca el tiempo solicitado (especificado en milisegundos) en el video. Si el video está en pausa, la búsqueda no actualizará el cuadro que se muestra. El jugador puede saltar solo a cuadros clave en el video, por lo que buscar tiempos que difieran en intervalos cortos puede que no se muevan a cuadros diferentes.

`Start()`
: Inicia la reproducción del video.

YandexTranslate

Use este componente para traducir palabras y oraciones entre diferentes idiomas. Este componente necesita acceso a Internet, ya que solicitará traducciones al servicio Yandex.Translate. Especifique el idioma de origen y de destino en el formulario fuente-objetivo utilizando dos códigos de idioma de letras. Entonces, "en-es" se traducirá de inglés a español, mientras que "es-ru" se traducirá de español a ruso. Si omite el idioma de origen, el servicio intentará detectar el idioma de origen. Entonces, al proporcionar solo "es" intentará detectar el idioma de origen y traducirlo al español.

Este componente es impulsado por el servicio de traducción Yandex. Consulte http://api.yandex.com/translate/ para obtener más información, incluida la lista de idiomas disponibles y los significados de los códigos de idioma y estado.

Nota: la traducción se realiza de forma asíncrona en segundo plano. Cuando se completa la traducción, se desencadena el evento "GotTranslation".

Propiedades

ninguna

Eventos

`GotTranslation(text responseCode, text translation)`
: Evento desencadenado cuando el servicio Yandex.Translate devuelve el texto traducido. Este evento también proporciona un código de respuesta para el manejo de errores. Si el código de respuesta no es 200, entonces algo salió mal con la llamada y la traducción no estará disponible.

Métodos

`RequestTranslation(text languageToTranslateTo, text textToTranslate)`
: Al proporcionar un idioma de destino para traducir a (por ejemplo, 'es' para español, 'en' para inglés o 'ru' para ruso) y una palabra o frase para traducir, este método solicitará una traducción al Yandex. Traducir servicio. Una vez que el servicio externo traduzca el texto, se ejecutará el evento GotTranslation. Nota: Yandex.Translate intentará detectar el idioma de origen. También puede especificar anteponerlo a la traducción del idioma. Es decir, es-ru especificará la traducción de español a ruso.

Componentes de dibujo y animación - App Inventor para Android

Tabla de contenido

Pelota

Un "sprite" redondo que se puede colocar en un `Canvas`, donde puede reaccionar a los toques y arrastra, interactuar con otros sprites (`ImageSprite`s y otros `Ball`) y el borde del lienzo, y moverse de acuerdo con sus valores de propiedad.
Por ejemplo, para tener un `Ball`movimiento de 4 píxeles hacia la parte superior de `Canvas`cada 500 milisegundos (medio segundo), establecería la `Speed`propiedad en 4 [píxeles], la `Interval`propiedad en 500 [milisegundos], la `Heading`propiedad en 90 [grados] y la `Enabled`propiedad a `True`. Estas y sus otras propiedades se pueden cambiar en cualquier momento.
La diferencia entre una bola y una `ImageSprite`es que este último pueda obtener su aspecto desde un archivo de imagen, mientras que la apariencia de una bola sólo puede cambiarse variando sus `PaintColor`y `Radius`propiedades.

Propiedades

`Enabled`
: Controla si el sprite se mueve cuando su velocidad no es cero.

`Heading`
: Devuelve el título del sprite en grados por encima del eje x positivo. Cero grados está hacia la derecha de la pantalla; 90 grados es hacia la parte superior de la pantalla.

`Interval`
: El intervalo en milisegundos en el que se actualiza la posición del sprite. Por ejemplo, si el intervalo es 50 y la velocidad es 10, entonces el sprite se moverá 10 píxeles cada 50 milisegundos.

`PaintColor`
`Radius`
`Speed`
: La velocidad a la que se mueve el sprite. El sprite mueve tantos píxeles en cada intervalo.

`Visible`
: Es cierto si el sprite es visible.

`X`
: La coordenada horizontal del borde izquierdo del sprite, que aumenta a medida que el sprite se mueve hacia la derecha.

`Y`
: La coordenada vertical de la parte superior del sprite, que aumenta a medida que el sprite se mueve hacia abajo.

`Z`
: Cómo se debe colocar el sprite en capas en relación con otros sprits, con capas de mayor numeración frente a las capas de menor número.

Eventos

`CollidedWith(component other)`
: Controlador para eventos CollidedWith, llamado cuando dos sprites colisionan. Tenga en cuenta que la comprobación de colisiones con un ImageSprite girado actualmente comprueba contra la posición no girada del sprite. Por lo tanto, la comprobación de colisión será inexacta para sprites altos estrechos o cortos que se rotan.

`Dragged(number startX, number startY, number prevX, number prevY, number currentX, number currentY)`
: Manejador de eventos arrastrados. En todas las llamadas, las coordenadas de inicio son donde se tocó la pantalla por primera vez, y las coordenadas "actuales" describen el punto final del segmento de línea actual. En la primera llamada dentro de un arrastre dado, las coordenadas "previas" son las mismas que las coordenadas iniciales; posteriormente, son las coordenadas "actuales" de la llamada anterior. Tenga en cuenta que el Sprite no se moverá a ninguna parte en respuesta al evento arrastrado a menos que se llame específicamente a MoveTo.

`EdgeReached(number edge)`
: Controlador de eventos llamado cuando el sprite llega a un borde de la pantalla. Si se llama a Bounce con ese borde, el sprite aparecerá rebotando fuera del borde que alcanzó. El borde aquí se representa como un número entero que indica una de las ocho direcciones norte (1), noreste (2), este (3), sureste (4), sur (-1), suroeste (-2), oeste (-3) y noroeste (-4).

`Flung(number x, number y, number speed, number heading, number xvel, number yvel)`
: Cuando se realiza un gesto de fling (deslizamiento rápido) en el sprite: proporciona la posición (x, y) del inicio de la aventura, relativa a la esquina superior izquierda del lienzo. También proporciona la velocidad (píxeles por milisegundo) y el rumbo (0-360 grados) de la aventura, así como las componentes de velocidad x y velocidad x del vector de la aventura.

`NoLongerCollidingWith(component other)`
: Evento que indica que un par de sprites ya no están colisionando.

`TouchDown(number x, number y)`
: Cuando el usuario comienza a tocar el sprite (coloca el dedo en el sprite y lo deja allí): proporciona la posición (x, y) del toque, relativa a la esquina superior izquierda del lienzo

`TouchUp(number x, number y)`
: Cuando el usuario deja de tocar el sprite (levanta el dedo después de un evento TouchDown): proporciona la posición (x, y) del toque, relativa a la esquina superior izquierda del lienzo

`Touched(number x, number y)`
: Cuando el usuario toca el sprite y luego levanta el dedo de forma inmediata: proporciona la posición (x, y) del toque, relativa a la esquina superior izquierda del lienzo

Métodos

`Bounce(number edge)`
: Hace que este sprite rebote, como si fuera una pared. Para el rebote normal, el argumento de borde debe ser el que devuelve EdgeReached.

`boolean CollidingWith(component other)`
: Indica si se ha registrado una colisión entre este sprite y el sprite pasado.

`MoveIntoBounds()`
: Mueve al sprite hacia atrás dentro de los límites si parte de él se extiende fuera de los límites, sin tener ningún efecto de lo contrario. Si el sprite es demasiado ancho para caber en el lienzo, este alinea el lado izquierdo del sprite con el lado izquierdo del lienzo. Si el sprite es demasiado alto para caber en el lienzo, este alinea el lado superior del sprite con el lado superior del lienzo.

`MoveTo(number x, number y)`
: Mueve el sprite de modo que su esquina superior izquierda se encuentre en las coordenadas xey especificadas.

`PointInDirection(number x, number y)`
: Gira el sprite para señalar hacia el punto con coordenadas como (x, y).

`PointTowards(component target)`
: Gira el sprite para apuntar hacia un sprite objetivo designado. El nuevo encabezado será paralelo a la línea que une los puntos centrales de los dos sprites.

Lona

Un panel rectangular táctil bidimensional en el que se puede dibujar y los objetos se pueden mover.

El `BackgroundColor`, `PaintColor`, `BackgroundImage`, `Width`, y `Height`del lienzo se puede configurar ya sea en el diseñador o en el Editor de bloques. El `Width`y `Height`se miden en píxeles y deben ser positivos.

Cualquier ubicación en el Lienzo se puede especificar como un par de valores (X, Y), donde

- X es la cantidad de píxeles del borde izquierdo del Lienzo
- Y es la cantidad de píxeles del borde superior del lienzo

.
Hay eventos para decir cuándo y dónde se ha tocado un Lienzo o se ha arrastrado un `Sprite` (`ImageSprite`o `Ball`). También hay métodos para dibujar puntos, líneas y círculos.

Propiedades

`BackgroundColor`
: El color del fondo del lienzo.

`BackgroundImage`
: El nombre de un archivo que contiene la imagen de fondo del lienzo

`FontSize`
: El tamaño de fuente del texto dibujado en el lienzo.

`Height`
`LineWidth`
: El ancho de las líneas dibujadas en el lienzo.

`PaintColor`
: El color en el que se dibujan las líneas

`TextAlignment`
: Determina la alineación del texto dibujado por DrawText () o DrawAngle () con respecto al punto especificado por ese comando: punto a la izquierda del texto, punto en el centro del texto o punto a la derecha del texto.

`Visible`
: Especifica si el componente debe ser visible en la pantalla. El valor es verdadero si el componente se muestra y falso si está oculto.

`Width`

Eventos

`Dragged(number startX, number startY, number prevX, number prevY, number currentX, number currentY, boolean draggedSprite)`
: Cuando el usuario hace un arrastre desde un punto (prevX, prevY) a otro (x, y). El par (startX, startY) indica dónde el usuario tocó la pantalla por primera vez, y "draggedSprite" indica si se está arrastrando un sprite.

`Flung(number x, number y, number speed, number heading, number xvel, number yvel, boolean flungSprite)`
: Cuando se realiza un gesto de arrojo (deslizamiento rápido) sobre el lienzo: proporciona la posición (x, y) del inicio de la aventura, relativa a la esquina superior izquierda del lienzo. También proporciona la velocidad (píxeles por milisegundo) y el rumbo (0-360 grados) de la aventura, así como las componentes de velocidad x y velocidad x del vector de la aventura. El valor "flungSprite" es verdadero si un sprite se encuentra cerca del punto de inicio del gesto de fling.

`TouchDown(number x, number y)`
: Cuando el usuario comienza a tocar el lienzo (coloca el dedo sobre el lienzo y lo deja allí): proporciona la posición (x, y) del toque, relativa a la esquina superior izquierda del lienzo

`TouchUp(number x, number y)`
: Cuando el usuario deja de tocar el lienzo (levanta el dedo después de un evento TouchDown): proporciona la posición (x, y) del toque, relativa a la esquina superior izquierda del lienzo

`Touched(number x, number y, boolean touchedSprite)`
: Cuando el usuario toca el lienzo e inmediatamente levanta el dedo: proporciona la posición (x, y) del toque, relativa a la esquina superior izquierda del lienzo. TouchedSprite es verdadero si el mismo toque también tocó un sprite, y de lo contrario es falso.

Métodos

`Clear()`
: Borra todo lo dibujado en este lienzo, pero no cualquier color o imagen de fondo.

`DrawCircle(number x, number y, number r)`
: Dibuja un círculo (rellenado) en las coordenadas dadas en el lienzo, con el radio dado.

`DrawLine(number x1, number y1, number x2, number y2)`
: Dibuja una línea entre las coordenadas dadas en el lienzo.

`DrawPoint(number x, number y)`
: Dibuja un punto en las coordenadas dadas en el lienzo.

`DrawText(text text, number x, number y)`
: Dibuja el texto especificado relativo a las coordenadas especificadas utilizando los valores de las propiedades FontSize y TextAlignment.

`DrawTextAtAngle(text text, number x, number y, number angle)`
: Dibuja el texto especificado comenzando en las coordenadas especificadas en el ángulo especificado utilizando los valores de las propiedades FontSize y TextAlignment.

`number GetBackgroundPixelColor(number x, number y)`
: Obtiene el color del punto especificado. Esto incluye el fondo y los puntos, líneas o círculos dibujados, pero no los sprites.

`number GetPixelColor(number x, number y)`
: Obtiene el color del punto especificado.

`text Save()`
: Guarda una imagen de este lienzo en el almacenamiento externo del dispositivo. Si ocurre un error, se llamará al evento ErrorOccurred de la pantalla.

`text SaveAs(text fileName)`
: Guarda una imagen de este lienzo en el almacenamiento externo del dispositivo en el archivo llamado fileName. fileName debe terminar con uno de .jpg, .jpeg o .png, que determina el tipo de archivo.

`SetBackgroundPixelColor(number x, number y, number color)`
: Establece el color del punto especificado. Esto difiere de DrawPoint al tener un argumento para el color.

ImageSprite

Un 'sprite' que se puede colocar en un `Canvas`, donde puede reaccionar a toques y drags, interactuar con otros sprites (`Ball`s y otros `ImageSprite`) y el borde del lienzo, y moverse de acuerdo con sus valores de propiedad. Su apariencia es la de la imagen especificada en su `Picture`propiedad (a menos que su `Visible`propiedad sea `False`.
Para tener un `ImageSprite`movimiento de 10 píxeles hacia la izquierda cada 1000 milisegundos (un segundo), por ejemplo, establecería la `Speed` propiedad en 10 [píxeles], la `Interval` propiedad en 1000 [milisegundos], la `Heading`propiedad en 180 [grados] y la `Enabled`propiedad a `True`. Un sprite cuya `Rotates`propiedad es `True`rotará su imagen como el sprite`Heading`cambios. La comprobación de colisiones con un sprite girado actualmente comprueba la posición no girada del sprite de modo que la comprobación de colisión será inexacta para sprites altos angostos o cortos que se rotan. Cualquiera de las propiedades de los sprites se puede cambiar en cualquier momento bajo el control del programa.

Propiedades

`Enabled`
: Controla si el sprite se mueve cuando su velocidad no es cero.

`Heading`
: Devuelve el título del sprite en grados por encima del eje x positivo. Cero grados está hacia la derecha de la pantalla; 90 grados es hacia la parte superior de la pantalla.

`Height`
`Interval`
: El intervalo en milisegundos en el que se actualiza la posición del sprite. Por ejemplo, si el intervalo es 50 y la velocidad es 10, entonces el sprite se moverá 10 píxeles cada 50 milisegundos.

`Picture`
: La imagen que determina la apariencia del sprite

`Rotates`
: Si es verdadero, la imagen del sprite gira para coincidir con el título del sprite. Si es falso, la imagen de sprite no gira cuando el sprite cambia de rumbo. El sprite gira alrededor de su punto central.

`Speed`
: La velocidad a la que se mueve el sprite. El sprite mueve tantos píxeles en cada intervalo.

`Visible`
: Es cierto si el sprite es visible.

`Width`

`X`

La coordenada horizontal del borde izquierdo del sprite, que aumenta a medida que el sprite se mueve hacia la derecha.

`Y`

La coordenada vertical de la parte superior del sprite, que aumenta a medida que el sprite se mueve hacia abajo.

`Z`

Cómo se debe colocar el sprite en capas en relación con otros sprits, con capas de mayor numeración frente a las capas de menor número.

Eventos

`CollidedWith(component other)`

Controlador para eventos CollidedWith, llamado cuando dos sprites colisionan. Tenga en cuenta que la comprobación de colisiones con un ImageSprite girado actualmente comprueba contra la posición no girada del sprite. Por lo tanto, la comprobación de colisión será inexacta para sprites altos estrechos o cortos que se rotan.

`Dragged(number startX, number startY, number prevX, number prevY, number currentX, number currentY)`

Manejador de eventos arrastrados. En todas las llamadas, las coordenadas de inicio son donde se tocó la pantalla por primera vez, y las coordenadas "actuales" describen el punto final del segmento de línea actual. En la primera llamada dentro de un arrastre dado, las coordenadas "previas" son las mismas que las coordenadas iniciales; posteriormente, son las coordenadas "actuales" de la llamada anterior. Tenga en cuenta que el Sprite no se moverá a ninguna parte en respuesta al evento arrastrado a menos que se llame específicamente a MoveTo.

`EdgeReached(number edge)`

Controlador de eventos llamado cuando el sprite llega a un borde de la pantalla. Si se llama a Bounce con ese borde, el sprite aparecerá rebotando fuera del borde que alcanzó. El borde aquí se representa como un número entero que indica una de las ocho direcciones norte (1), noreste (2), este (3), sureste (4), sur (-1), suroeste (-2), oeste (-3) y noroeste (-4).

`Flung(number x, number y, number speed, number heading, number xvel, number yvel)`

Cuando se realiza un gesto de fling (deslizamiento rápido) en el sprite: proporciona la posición (x, y) del inicio de la aventura, relativa a la esquina superior izquierda del lienzo. También proporciona la velocidad (píxeles por milisegundo) y el rumbo (0-360 grados) de la aventura, así como las componentes de velocidad x y velocidad x del vector de la aventura.

`NoLongerCollidingWith(component other)`

Evento que indica que un par de sprites ya no están colisionando.

`TouchDown(number x, number y)`

Cuando el usuario comienza a tocar el sprite (coloca el dedo en el sprite y lo deja allí): proporciona la posición (x, y) del toque, relativa a la esquina superior izquierda del lienzo

`TouchUp(number x, number y)`

Cuando el usuario deja de tocar el sprite (levanta el dedo después de un evento TouchDown): proporciona la posición (x, y) del toque, relativa a la esquina superior izquierda del lienzo

`Touched(number x, number y)`

Cuando el usuario toca el sprite y luego levanta el dedo de forma inmediata: proporciona la posición (x, y) del toque, relativa a la esquina superior izquierda del lienzo

Métodos

`Bounce(number edge)`

Hace que este sprite rebote, como si fuera una pared. Para el rebote normal, el argumento de borde debe ser el que devuelve EdgeReached.

`boolean CollidingWith(component other)`

Indica si se ha registrado una colisión entre este sprite y el sprite pasado.

`MoveIntoBounds()`

Mueve al sprite hacia atrás dentro de los límites si parte de él se extiende fuera de los límites, sin tener ningún efecto de lo contrario. Si el sprite es demasiado ancho para caber en el lienzo, este alinea el lado izquierdo del sprite con el lado izquierdo del lienzo. Si el sprite es demasiado alto para caber en el lienzo, este alinea el lado superior del sprite con el lado superior del lienzo.

`MoveTo(number x, number y)`

Mueve el sprite de modo que su esquina superior izquierda se encuentre en las coordenadas xey especificadas.

`PointInDirection(number x, number y)`

Gira el sprite para señalar hacia el punto con coordenadas como (x, y).

`PointTowards(component target)`
Gira el sprite para apuntar hacia un sprite objetivo designado. El nuevo encabezado será paralelo a la línea que une los puntos centrales de los dos sprites.

Componentes de Maps - App Inventor para Android

Tabla de contenido

Circulo

El componente Círculo visualiza un círculo de un radio dado, en metros, en una latitud y longitud. Appearnce del círculo puede ser personalizado utilizando propiedades tales como `FillColor`, `StrokeColor`, y `StrokeWidth`.
El componente Circle también se puede usar para implementar funciones como geofencing, un mecanismo donde la presencia del usuario dentro de un área se usa para desencadenar otros comportamientos. Al usar el `DistanceToPoint`método combinado con el LocationSensor , puede determinar si la ubicación de un usuario está dentro o fuera del círculo. Puede usar esta característica para activar acciones adicionales.

Propiedades

`Descripción`
Establece u obtiene la descripción que se muestra en la ventana de información. La ventana de información aparece cuando el usuario toca el círculo.

`Arrastrable`
Establece u obtiene si el usuario puede o no arrastrar una función de mapa. Se accede a esta función pulsando prolongadamente y luego arrastrando el círculo a una nueva ubicación.

`EnableInfobox`
Habilita o deshabilita la visualización de la ventana del cuadro de información cuando el usuario toca el círculo.

`Color de relleno`
Establece u obtiene el color utilizado para completar el círculo.

`Latitud`
Establece u obtiene la latitud del centro del círculo, en grados. Valores positivos que representan el norte del ecuador y valores negativos que representan al sur del ecuador. Para actualizar la latitud y la longitud simultáneamente, use el `SetLocation` método.

`Longitud`
Establece u obtiene la longitud del centro del círculo, en grados. Valores positivos que representan al este del meridiano principal y valores negativos que representan al oeste del meridiano principal. Para actualizar la latitud y la longitud simultáneamente, use el `SetLocation`método.

`Radio`
Establece u obtiene el radio del círculo, en metros.

`Color del trazo`
Establece u obtiene el color utilizado para delinear el círculo.

`Anchura del trazo`
Establece u obtiene el ancho del trazo utilizado para delinear el círculo.

`Título`
Establece u obtiene el título que se muestra en la ventana de información que aparece cuando el usuario hace clic en la función del mapa.

`Tipo`
Obtiene el tipo de la característica. Para Circle, esto siempre será "Circle",

`Visible`
Establece u obtiene si el componente debe ser visible en la pantalla. El valor es verdadero si el componente se muestra y falso si está oculto.

Eventos

`Hacer clic`

Se ejecuta cuando el usuario toca el círculo.

`Arrastrar`
Se ejecuta durante las operaciones de arrastre.

`LongClick`
Se ejecuta después de que el usuario hace clic largo en el círculo, pero no desencadena un arrastre. Tenga en cuenta que este evento solo se activará si `Draggable`es falso.

`StartDrag`
Se ejecuta antes de que comience una operación de arrastre. Use esto para guardar la posición actual del círculo, por ejemplo.

`StopDrag`
Se ejecuta después de una operación de arrastre completa. Use esto para guardar la nueva posición del círculo, por ejemplo.

Métodos

`número DistanceToFeature (componente mapFeature, boolean centroids)`
Calcula la distancia entre el Círculo y la característica de mapa dada. Si `centroids`es verdadero, el cálculo se hace entre los centroides de las dos características. De lo contrario, la distancia se calculará entre las dos características en función de los puntos más cercanos. Además, cuando `centroids`es falso, este método devolverá 0 si el círculo se cruza o contiene el `mapFeature`. Si ocurre un error, este método devolverá -1.

`número DistanceToPoint (latitud, longitud, centroides booleanos)`
Calcula la distancia entre el Círculo y la latitud y longitud dadas. Si `centroids`es verdadero, la distancia se calcula desde el centro del círculo hasta el punto dado. De lo contrario, la distancia se calcula desde el punto más cercano en el círculo hasta el punto dado. Además, este método devolverá 0 si `centroids`es falso y el punto está en el círculo. Si ocurre un error, se devolverá -1.

`HideInfobox`
Oculta el cuadro de información del círculo si está visible. De lo contrario, no se realiza ninguna acción.

`SetLocation (latitud, longitud del número)`
Mueve el centro del círculo a la latitud y longitud especificadas. Este método es más eficiente que establecer la latitud y la longitud por separado.

`ShowInfobox`
Muestra el cuadro de información para el círculo si no está visible. De lo contrario, este método no tiene ningún efecto. Este método se puede usar para mostrar el cuadro de información aunque `EnableInfobox`sea falso.

FeatureCollection

Una FeatureCollection agrupa una o más entidades de mapa juntas. Cualquier evento que ocurra dentro de una característica en la colección también activará el evento correspondiente en el componente de recopilación. FeatureCollections se puede cargar desde recursos de ejercicio para poblar Maps con contenido. GeoJSON es el único formato admitido en este momento.

Propiedades

`Caracteristicas`
Devuelve una lista de características presentes en la colección de características, si corresponde.

`Características de GeeoJSON`
Rellena la colección de características de una cadena que contiene contenido GeoJSON. Dado el tamaño de tales cadenas, se recomienda cargar la colección de características de los activos o la web utilizando la propiedad Fuente.

`Fuente`
La fuente del contenido para la colección de características, como un nombre de archivo o una URL.

`Visible`
Especifica si el componente debe ser visible en la pantalla. El valor es verdadero si el componente se muestra y falso si está oculto.

Eventos

`FeatureClick (función del componente)`
Cuando se hace clic en una característica, la colección de características que la contiene (si existe) también recibirá un `FeatureClick`evento. El parámetro de función indica en qué función secundaria se hizo clic.

`FeatureDrag (característica del componente)`
Cuando se arrastra una característica, la colección principal también recibirá un `FeatureDrag` evento. El parámetro de característica indica qué característica secundaria se arrastró.

`FeatureLongClick (característica del componente)`
Cuando se hace clic largo en una característica, la colección principal también recibirá un `FeatureLongClick`evento. El parámetro de característica indica en qué función secundaria se hizo clic durante mucho tiempo.

`FeatureStartDrag (característica del componente)`

Cuando el usuario comienza a arrastrar una característica, la colección principal también recibirá un `FeatureStartDrag`evento. El parámetro de característica indica qué característica secundaria se arrastró.

`FeatureStopDrag (característica del componente)`

Cuando el usuario deja de arrastrar una característica, la colección principal también recibirá un `FeatureStopDrag`evento. El parámetro de característica indica qué característica secundaria se arrastró.

`GotFeatures (url de texto, funciones de lista)`

El evento GotFeatures se ejecuta cuando una colección de características se lee con éxito del dado `url`. El `features`parámetro será una lista de descripciones de características que se pueden convertir en componentes utilizando el `FeatureFromDescription` método.

`LoadError (url de texto, funciones de lista)`

El evento LoadError se ejecuta cuando se produce un error al procesar un documento de colección de características en el momento dado `url`. El `responseCode`parámetro contendrá un código de estado HTTP y el `errorMessage`parámetro contendrá un mensaje de error detallado.

Métodos

`cualquier FeatureFromDescription (descripción de la lista)`

Devuelve un nuevo componente basado en la descripción proporcionada. Si hay un error en las propiedades, como datos con formato incorrecto, el método devolverá el texto que describe el error. Use el `is-text?` bloque para probar si el resultado es un mensaje de error. es un error o no Las propiedades de las características se convierten en propiedades de App Inventor utilizando la siguiente asignación insensible a mayúsculas y minúsculas:

- descripción → Descripción
- arrastrable → Arrastrable
- infobox → EnableInfobox
- llenar → FillColor
- imagen → ImageAsset
- trazo → StrokeColor
- stroke-width → StrokeWidth
- título → Título
- visible → Visible

`LoadFromURL (url de texto)`

Llame a este método para cargar una descripción de GeoJSON de una colección de características desde una URL (incluidas las URL de los archivos). Si tiene éxito, el conjunto de características administradas por la colección de características será reemplazado por las nuevas características y se ejecutará el evento LoadFeatureCollection. Si se produce un error, se ejecutará el evento ErrorLoadingFeatureCollection.

LineString

LineString es un componente para dibujar una secuencia abierta y continua de líneas en un Mapa. Para agregar nuevos puntos a LineString, arrastre el punto medio de cualquier segmento fuera de la línea para introducir un nuevo vértice. Mueva un vértice haciendo clic y arrastrando el vértice a una nueva ubicación. Al hacer clic en un vértice se eliminará el vértice.

Propiedades

`Descripción`

La descripción que se muestra en la ventana de información que aparece cuando el usuario hace clic en la función del mapa.

`Arrastrable`

Establece u obtiene si el usuario puede o no arrastrar una cadena de líneas presionando prolongadamente y luego arrastrándola a una nueva ubicación.

`EnableInfobox`

Habilite o deshabilite la ventana de la ventana de información cuando el usuario toca la función.

`Puntos`

La lista de puntos, como pares de latitudes y longitudes, en LineString.

`PointsFromString`

Una cadena de puntos codificada GeoJSON para poblar LineString. La edición de LineString en el diseñador actualizará esta propiedad.

`Color del trazo`

El color de pintura utilizado para delinear la función de mapa.

`Anchura del trazo`

El ancho del trazo utilizado para delinear la función de mapa.

`Título`

El título que se muestra en la ventana de información que aparece cuando el usuario hace clic en la función del mapa.

`Tipo`

Obtiene el tipo de la característica. Para LineString, esto siempre será "LineString",

`Visible`
Especifica si el componente debe ser visible en la pantalla. El valor es verdadero si el componente se muestra y falso si está oculto.

Eventos

`Hacer clic`
Se ejecuta cuando el usuario toca ligeramente o muy cerca de la cadena de línea.

`Arrastrar`
Se ejecuta durante una operación de arrastre.

`LongClick`
Se ejecuta después de que el usuario hace clic largo en la cadena de línea, pero no desencadena un arrastre (dentro de un umbral dado).

`StartDrag`
Se ejecuta inmediatamente después de que el usuario comienza una operación de arrastre pero antes de cualquier evento de Arrastrar.

`StopDrag`
Se ejecuta después de que el usuario libera LineString de una operación de arrastre.

Métodos

`número DistanceToFeature (componente mapFeature, boolean centroids)`
Calcula la distancia entre LineString y el dado `mapFeature`. Si `centroids`es verdadero, el cálculo se hace entre los centroides de las dos características. Si es falso, la distancia se calculará entre las dos características en función de los puntos de cierre. Si la cadena lineal interseca el `mapFeature`, este método devolverá 0. Si se produce un error, se devolverá -1.

`número DistanceToPoint (latitud, longitud, centroides booleanos)`
Calcula la distancia entre LineString y el dado `latitude` y `longitude`. Si `centroids`es verdadero, el cálculo se realiza entre el punto medio ponderado de LineString hasta el punto. Si es falso, la distancia se calcula desde el punto más cercano en el LineString hasta el punto. Si el punto está en LineString, este método devolverá 0. Si se produce un error -1 se devolverá.

`HideInfobox`
Oculta el cuadro de información de la cadena de líneas si está visible. De lo contrario, este método no tiene ningún efecto.

`ShowInfobox`
Muestra el cuadro de información para la cadena de línea si no está visible. De lo contrario, este método no tiene ningún efecto. Este método se puede usar para mostrar el cuadro de información aunque `EnableInfobox`sea falso.

Mapa

Un contenedor bidimensional que muestra mosaicos de mapa en segundo plano y permite que múltiples elementos de Marker identifiquen puntos en el mapa. Los mosaicos de mapas son suministrados por los colaboradores de OpenStreetMap y el Servicio Geológico de los Estados Unidos.

El componente Map proporciona tres utilidades para manipular sus límites dentro de App Inventor. Primero, un mecanismo de bloqueo permite que el mapa se mueva con relación a otros componentes en la pantalla. En segundo lugar, cuando se desbloquea, el usuario puede desplazar el mapa a cualquier ubicación. En esta nueva ubicación, se puede presionar el botón "Establecer límite inicial" para guardar las coordenadas actuales del mapa en sus propiedades. Por último, si el mapa se mueve a una ubicación diferente, por ejemplo para agregar marcadores fuera de la pantalla, entonces el botón "Restablecer mapa a límites iniciales" se puede utilizar para volver a centrar el mapa en la ubicación de inicio.

Propiedades

`Cuadro delimitador`
Establece u obtiene el límite actual para la vista dibujada del mapa. El valor es una lista de listas que contiene las coordenadas noroeste y sureste de la vista actual en el formulario `((North West) (South East))`.

`CenterFromString`
Establece el centro del mapa a partir de una cadena dada de "latitud, longitud". Esto se usa principalmente para poblar el centro del mapa del diseñador. Vea también el `PanTo`método para animar un cambio al centro del Mapa.

`EnablePan`
Habilita o deshabilita la capacidad del usuario para mover el mapa.

`EnableRotation`
Activa o desactiva el gesto de rotación de dos dedos para rotar el mapa.

`EnableZoom`
Habilita o deshabilita el gesto de pellizco de dos dedos para acercar o alejar el mapa.

`Caracteristicas`

Establece u obtiene una lista de características presentes en el Mapa. Establecer esto en una lista vacía borrará el mapa.

`Altura`
Establece u obtiene la altura del mapa.

`AlturaPercent`
Establece la altura del Mapa en un porcentaje de la Pantalla.

`Latitud`
Obtiene la latitud del centro del mapa. Para cambiar la latitud, usa el `PanTo`método.

`LocationSensor`
Utiliza el LocationSensor proporcionado para los datos de ubicación del usuario en lugar del proveedor de ubicación incorporado.

`Longitud`
Obtiene la longitud del centro del mapa. Para cambiar la longitud, usa el `PanTo`método.

`Tipo de mapa`
Establece u obtiene la capa de mosaico utilizada para dibujar el fondo del mapa. Predeterminado a Roads. Los valores válidos son 1 (Carreteras), 2 (Aérea) o 3 (Terreno). Las capas de camino son proporcionadas por OpenStreetMap y las capas aéreas y de terreno son proporcionadas por el Servicio Geológico de los EE. UU.

`ShowCompass`
Muestra u oculta una superposición de brújula en el mapa. La brújula se rotará en función de la orientación del dispositivo si hay una brújula digital en el hardware.

`ShowUser`
Muestra u oculta un ícono que indica la ubicación actual del usuario en el Mapa. La disponibilidad y precisión de esta función dependerá de si el usuario tiene habilitados los servicios de localización y qué proveedores de ubicación están disponibles.

`ShowZoom`
Muestra u oculta los botones de zoom nativos de Android para permitir al usuario acercar o alejar el mapa. Esto se puede usar en lugar del gesto de pellizcar para hacer zoom con dos dedos. Los controles de zoom no se comportan correctamente cuando la propiedad de Tamaño en la Pantalla1 está establecida en Fijo. Cambie al uso del dimensionamiento receptivo como una solución alternativa hasta que se solucione este error.

`UserLatitude`
Devuelve la latitud del usuario si ShowUser está habilitado.

`UserLongitude`
Devuelve la longitud del usuario si ShowUser está habilitado.

`Visible`
Establece u obtiene si el mapa está visible.

`Anchura`
Establece u obtiene el ancho del mapa.

`WidthPercent`
Establece el ancho del mapa a un porcentaje de la pantalla.

`Nivel de zoom`
Obtiene o establece el nivel de zoom para el mapa. Los valores válidos van desde 1-20. No todas las capas de mosaico admitirán cada nivel de zoom en cada ubicación. Por ejemplo, es probable que la fotografía aérea detallada no esté disponible para los mosaicos en el medio del océano o en los polos. Es probable que se produzcan los niveles de zoom más altos en los principales centros de las ciudades debido a la cantidad de datos detallados disponibles.

Eventos

`BoundsChange`
Se ejecuta cuando el usuario cambia los límites del mapa, ya sea al acercar, alternar o girar la vista.

`DoubleTapAtPoint (latitud, longitud numérica)`
Se ejecuta cuando el usuario toca dos veces en un punto del mapa. `latitude` e `longitude`indique la ubicación del evento tap en las coordenadas del mapa. Este evento puede ser seguido por un `ZoomChange`evento si los gestos de acercamiento están habilitados y el mapa no está en el nivel de zoom más alto posible.

`FeatureClick (función del componente)`
Cuando se hace clic en una característica, el mapa principal también recibirá un `FeatureClick` evento. El `feature`parámetro indica en qué función secundaria se hizo clic.

`FeatureDrag (característica del componente)`
Cuando se arrastra una característica, el mapa principal también recibirá un `FeatureDrag`evento. El `feature`parámetro indica qué característica secundaria fue arrastrada.

`FeatureLongClick (característica del componente)`
Cuando se hace clic largo en una característica, el mapa principal también recibirá un `FeatureLongClick` evento. El `feature`parámetro indica en qué función secundaria se hizo clic durante mucho tiempo.

`FeatureStartDrag (característica del componente)`
Cuando el usuario comienza a arrastrar una característica, el mapa principal también recibirá un `FeatureStartDrag`evento. El `feature`parámetro indica qué característica secundaria fue arrastrada.

`FeatureStopDrag (característica del componente)`

Cuando el usuario deja de arrastrar una característica, el mapa principal también recibirá un `FeatureStopDrag`evento. El `feature`parámetro indica qué característica secundaria fue arrastrada.

`GotFeatures (url de texto, funciones de lista)`

El `GotFeatures`evento se ejecuta después de una llamada para `LoadFromURL`leer correctamente la descripción de la función `url`. El `features`parámetro será una lista de descripciones de características que se pueden convertir en componentes utilizando el `FeatureFromDescription` método.

`InvalidPoint (mensaje de texto)`

Se ejecuta cuando el programa encuentra un punto no válido al procesar datos geográficos. Los puntos se consideran inválidos cuando la latitud o longitud del punto está fuera del rango aceptable ([-90, 90] y [-180, 180], respectivamente). El `message`parámetro contendrá una explicación del error.

`LoadError (url de texto, número responseCode, texto errorMessage)`

El evento LoadError se ejecuta cuando el procesamiento de un documento de colección de características `url`produce un error. El `responseCode`parámetro contendrá un código de estado HTTP y el `errorMessage`parámetro contendrá un mensaje de error detallado.

`LongPressAtPoint (latitud, longitud numérica)`

Se ejecuta cuando el usuario pulsa un punto en el mapa. `Latitude` e `longitude`indique la ubicación de la pulsación larga en las coordenadas del mapa. Tenga en cuenta que este evento no se activará si `EnablePan`es verdadero, ya que una pulsación larga ocasiona un evento de panoramización en su lugar.

`Listo`

Se ejecuta cuando el mapa se ha iniciado y está listo para su uso.

`TapAtPoint (latitud, longitud del número)`

Se ejecuta cuando el usuario toca un punto en el mapa. La ubicación girada se informará en las coordenadas del mapa a través de los parámetros `latitude`y `longitude`.

`ZoomChange`

Se ejecuta cuando el usuario cambia el nivel de zoom, como mediante un gesto de pellizco o tocando dos veces.

Métodos

`componente CreateMarker (latitud, longitud numérica)`

Crea un nuevo marcador en el mapa en el dado `latitude` y `longitude`. El marcador se puede manipular usando los bloques "cualquier componente".

`cualquier FeatureFromDescription (descripción de la lista)`

Devuelve un nuevo componente basado en la descripción proporcionada. Si hay un error en las propiedades, como datos con formato incorrecto, el método devolverá el texto que describe el error. Usa el `is-text?` bloque para probar si obtienes un error. Las propiedades de característica convertidas en propiedades de App Inventor utilizan el siguiente mapeo insensible de mayúsculas y minúsculas:

- descripción → Descripción
- arrastrable → Arrastrable
- infobox → EnableInfobox
- llenar → FillColor
- imagen → ImageAsset
- trazo → StrokeColor
- stroke-width → StrokeWidth
- título → Título
- visible → Visible

`LoadFromURL (url de texto)`

Llame a este método para cargar una colección de características desde una URL (incluidas las URL de los archivos). Si el evento es exitoso, las descripciones de las características se pasan como una lista al `GotFeatures`evento. Si falla, el `LoadError`evento se ejecutará. En este momento, GeoJSON es el único formato compatible.

`PanTo (latitud, longitud, número de zoom)`

Mueve el centro del mapa a lo dado `(Latitude Longitude)`y se acerca al dado `zoom`. El movimiento está animado.

`Guardar (ruta de texto)`

Guarda descripciones del contenido del mapa en la ruta determinada. Actualmente, esto solo guardará las características usando el formato GeoJSON.

Marcador

El componente Marcador indica puntos en un Mapa, como edificios o puntos de interés. Los marcadores se pueden personalizar de muchas maneras, como el uso de imágenes personalizadas de los recursos de la aplicación. Los marcadores también se pueden crear dinámicamente utilizando el `CreateMarker`método y configurados utilizando los bloques "Cualquier componente".

Propiedades

`AnchorHorizontal`

Establece u obtiene el desplazamiento horizontal del centro del marcador en relación con su imagen. Los valores válidos son 1 (izquierda), 2 (derecha), 3 (centro)

`AnchorVertical`
: Establece u obtiene el desplazamiento vertical del centro del marcador en relación con su imagen. Los valores válidos son 1 (arriba), 2 (centro), 3 (abajo).

`Descripción`
: Establece u obtiene la descripción que se muestra en la ventana de información que aparece cuando el usuario toca el marcador.

`Arrastrable`
: La propiedad Draggable se utiliza para controlar si el usuario puede o no arrastrar el marcador al presionar y arrastrar el marcador a una nueva ubicación.

`EnableInfobox`
: Habilita o deshabilita la visualización de la ventana del cuadro de información cuando el usuario toca el marcador.

`Color de relleno`
: Establece u obtiene el color utilizado para completar el marcador. Esta propiedad solo se aplica a los marcadores que usan activos de imágenes vectoriales, incluido el ícono predeterminado.

`Altura`
: Establece u obtiene la altura del marcador, en píxeles.

`AlturaPercent`
: Establece la altura del marcador como un porcentaje de la altura de la pantalla.

`ImageAsset`
: Establece u obtiene la imagen que se muestra para el marcador. Si se establece en la cadena vacía `""`, se usará el ícono de marcador predeterminado.

`Latitud`
: Establece u obtiene la latitud del marcador, en grados, con valores positivos que representan al norte del ecuador y valores negativos que representan al sur del ecuador. Para actualizar la latitud y la longitud simultáneamente, use el `SetLocation` método.

`Longitud`
: Establece u obtiene la longitud del marcador, en grados, con valores positivos que representan al este del meridiano principal y valores negativos que representan al oeste del meridiano principal. Para actualizar la latitud y la longitud simultáneamente, use el `SetLocation`método.

`Color del trazo`
: Establece u obtiene el color utilizado para delinear el marcador.

`Anchura del trazo`
: Establece u obtiene el ancho del trazo utilizado para delinear el marcador.

`Título`
: Establece u obtiene el título que se muestra en la ventana de información que aparece cuando el usuario hace clic en el marcador.

`Tipo`
: Obtiene el tipo de la característica. Para Marker, este siempre será "Marker",

`Visible`
: Establece u obtiene si el componente debe ser visible en la pantalla. El valor es verdadero si el componente se muestra y falso si está oculto.

`Anchura`
: Establece u obtiene el ancho del marcador, en píxeles.

`WidthPercent`
: Establece el ancho del marcador como un porcentaje del ancho de la pantalla.

Eventos

`Hacer clic`
: Se ejecuta cuando el usuario toca el marcador.

`Arrastrar`
: Se ejecuta continuamente mientras un usuario está arrastrando el marcador.

`LongClick`
: Se ejecuta cuando el usuario hace clic largo en el marcador, pero no desencadena un arrastre. Tenga en cuenta que este evento solo se ejecutará si `Draggable`es falso.

`StartDrag`
: Se ejecuta antes de que comience una operación de arrastre.

`StopDrag`
: Se ejecuta después de una operación de arrastre completa.

Métodos

`número BearingToFeature (componente mapFeature, boolean centroids)`
: Devuelve el rumbo del marcador al dado `mapFeature`, en grados desde el norte. Si el `centroids`paremter es verdadero, el rumbo estará en el centro de la función del mapa. De lo contrario, el rumbo se calculará hasta el punto en la característica más cercana al Marcador.

`número BearingToPoint (latitud, longitud, centroides booleanos)`
: Devuelve el rumbo desde el Marcador a la latitud y longitud especificadas, en grados desde el norte derecho.

`número DistanceToFeature (componente mapFeature, boolean centroids)`

Calcula la distancia entre el Marcador y el dado `mapFeature`. Si `centroids`es verdadero, el cálculo se hace entre los centroides de las dos características. De lo contrario, la distancia se calculará entre las dos características en función de los puntos más cercanos. Además, cuando `centroids`es falso, este método devolverá 0 si el marcador se cruza o contiene el `mapFeature`. Si ocurre un error, este método devolverá -1.

`número DistanceToPoint (latitud, longitud, centroides booleanos)`

Calcula la distancia entre el Marcador y la latitud y longitud dadas. Si `centroids`es verdadero, la distancia se calcula desde el centro del círculo hasta el punto dado. De lo contrario, la distancia se calcula desde el punto más cercano en el marcador hasta el punto dado. Si ocurre un error, se devolverá -1.

`HideInfobox`

Oculta el cuadro de información del círculo si está visible. De lo contrario, no se tomará ninguna medida.

`SetLocation (latitud, longitud del número)`

Mueve el centro del círculo a la latitud y longitud especificadas. Este método es más eficiente que establecer la latitud y la longitud por separado.

`ShowInfobox`

Muestra el cuadro de información para el círculo si no está visible. De lo contrario, no se tomará ninguna medida. Este método se puede usar para mostrar el cuadro de información incluso si EnableInfobox es falso.

Polígono

El polígono encierra un área bidimensional arbitraria en un mapa. Los polígonos se pueden usar para dibujar un perímetro, como un campus, una ciudad o un país. Los polígonos comienzan como triángulos básicos. Se pueden crear nuevos vértices arrastrando el punto medio de un polígono lejos del borde. Al hacer clic en un vértice se eliminará el vértice, pero debe existir un mínimo de 3 vértices en todo momento.

Propiedades

`Descripción`

Establece u obtiene la descripción que se muestra en la ventana de información que aparece cuando el usuario toca el polígono.

`Arrastrable`

Establece u obtiene si el usuario puede o no arrastrar el polígono pulsando prolongadamente y luego arrastrándolo a una nueva ubicación.

`EnableInfobox`

Habilita o deshabilita la visualización de la ventana de infobox cuando el usuario toca el polígono.

`Color de relleno`

Establece u obtiene el color utilizado para completar el polígono.

`HolePoints`

Establece u obtiene las listas de puntos que comprenden los agujeros del polígono. Los puntos se dan como `(Latitude Longitude)`pares y se deben dar en sentido antihorario.

`HolePointsFromString`

Establece la lista de puntos de agujero de una cadena en formato GeoJSON.

`Puntos`

Establece u obtiene la lista de puntos que componen el polígono. Los puntos se dan en `(Latitude Longitude)`pares y deben darse en el sentido de las agujas del reloj.

`PointsFromString`

Establece la lista de puntos de una cadena en formato GeoJSON.

`Color del trazo`

Establece u obtiene el color utilizado para delinear el polígono.

`Anchura del trazo`

Establece u obtiene el ancho del trazo utilizado para delinear el polígono.

`Título`

Establece u obtiene el título que se muestra en la ventana de información que aparece cuando el usuario hace clic en la función del mapa.

`Tipo`

Obtiene el tipo de la característica. Para Polygon, este siempre será "Polígono",

`Visible`

Establece u obtiene si el componente debe ser visible en la pantalla. El valor es verdadero si el componente se muestra y falso si está oculto.

Eventos

`Hacer clic`

Se ejecuta cuando el usuario toca el polígono.

`Arrastrar`

Se ejecuta continuamente mientras un usuario está arrastrando el polígono.

`LongClick`

Se ejecuta cuando el usuario hace clic largo en el polígono pero no desencadena un arrastre. Tenga en cuenta que este evento solo se ejecutará si `Draggable`es falso.

`StartDrag`
: Se ejecuta antes de que comience una operación de arrastre.

`StopDrag`
: Se ejecuta después de una operación de arrastre completa.

Métodos

`lista Centroide`
: Devuelve el centroide del polígono como una lista del formulario `(Latitude Longitude)`.

`número DistanceToFeature (componente mapFeature, boolean centroid)`
: Calcula la distancia entre el polígono y el mapFeature dado. Si centroides es verdadero, el cálculo se realiza entre los centroides de las dos características. De lo contrario, la distancia se calculará entre las dos características en función de los puntos más cercanos. Además, si centroides es falso, este método devolverá 0 si el polígono intersecta o contiene el mapFeature. Si ocurre un error, este método devolverá -1.

`número DistanceToPoint (latitud del número, longitud del número, centroide booleano)`
: Calcula la distancia entre el polígono y la latitud y longitud dadas. Si centroides es verdadero, la distancia se calcula desde el centro del polígono hasta el punto dado. De lo contrario, la distancia se calcula desde el punto más cercano en el polígono hasta el punto dado. Además, este método devolverá 0 si centroides es falso y el punto está en el polígono. Si ocurre un error, se devolverá -1.

`HideInfobox`
: Oculta el cuadro de información del polígono si está visible. De lo contrario, este método no tiene ningún efecto.

`ShowInfobox`
: Muestra el cuadro de información del polígono si no está visible. De lo contrario, este método no tiene ningún efecto. Este método se puede usar para mostrar el cuadro de información aunque `EnableInfobox`sea falso.

Rectángulo

Los rectángulos son polígonos con latitudes y longitudes fijas para los límites norte, sur, este y oeste. Mover un vértice del rectángulo actualiza los bordes apropiados en consecuencia.

Propiedades

`Descripción`
: Establece u obtiene la descripción que se muestra en la ventana de información que aparece cuando el usuario toca el rectángulo.

`Arrastrable`
: Establece u obtiene si el usuario puede o no arrastrar una entidad de mapa al presionar prolongadamente y luego arrastrar el rectángulo a una nueva ubicación.

`EastLongitude`
: Establece u obtiene la longitud que delimita el rectángulo en el este. Rango: [-180, 180]

`EnableInfobox`
: Habilita o deshabilita la visualización de la ventana del cuadro de información cuando el usuario toca el rectángulo.

`Color de relleno`
: Establece u obtiene el color utilizado para rellenar el rectángulo.

`NorthLatitude`
: Establece u obtiene la latitud, en grados, que delimita el rectángulo en el norte. Rango: [-90, 90]

`SouthLatitude`
: Establece u obtiene la latitud, en grados, que delimita el rectángulo en el sur. Rango: [-90, 90]

`Color del trazo`
: Establece u obtiene el color utilizado para delinear el rectángulo.

`Anchura del trazo`
: Establece u obtiene el ancho del trazo utilizado para delinear el rectángulo.

`Título`
: Establece u obtiene el título que se muestra en la ventana de información que aparece cuando el usuario hace clic en el rectángulo.

`Tipo`
: Obtiene el tipo de la característica. Para Rectángulo, esto siempre será "Rectángulo",

`Visible`
: Establece u obtiene si el componente debe ser visible en la pantalla. El valor es verdadero si el componente se muestra y falso si está oculto.

`WestLongitude`
: Establece u obtiene la longitud, en grados, incrustando el rectángulo en el oeste. Rango: [-180, 180]

Eventos

`Hacer clic`
: Se ejecuta cuando el usuario toca el rectángulo.

`Arrastrar`
: Se ejecuta continuamente mientras un usuario está arrastrando el rectángulo.

`LongClick`
: Se ejecuta cuando el usuario hace clic largo en el rectángulo pero no desencadena un arrastre. Tenga en cuenta que este evento solo se ejecutará si `Draggable`es falso.

`StartDrag`
: Se ejecuta antes de que comience una operación de arrastre.

`StopDrag`
: Se ejecuta después de una operación de arrastre completa.

Métodos

`lista límites`
: Devuelve el cuadro delimitador del rectángulo en el formato `((North West) (South East))`.

`lista Centro`
: Devuelve el centro del Rectángulo como una lista del formulario `(Latitude Longitude)`.

`número DistanceToFeature (componente mapFeature, boolean centroids)`
: Calcula la distancia entre el Rectangle y el mapFeature dado. Si centroides es verdadero, el cálculo se realiza entre los centroides de las dos características. De lo contrario, la distancia se calculará entre las dos características en función de los puntos más cercanos. Además, cuando centroides es falso, este método devolverá 0 si el rectángulo se cruza o contiene el `mapFeature`. Si ocurre un error, este método devolverá -1.

`número DistanceToPoint (latitud, longitud, centroides booleanos)`
: Calcula la distancia entre el Rectángulo y la latitud y longitud dadas. Si centroides es verdadero, la distancia se calcula desde el centro del rectángulo hasta el punto dado. De lo contrario, la distancia se calcula desde el punto más cercano en el rectángulo hasta el punto dado. Además, este método devolverá 0 si centroides es falso y el punto está en el rectángulo. Si ocurre un error, se devolverá -1.

`HideInfobox`
: Oculta el cuadro de información del rectángulo si está visible. De lo contrario, este método no tiene ningún efecto.

`SetCenter (latitud, longitud numérica)`
: Mueve el rectángulo de modo que se centre en el dado `latitude` y `longitude`al intentar mantener el ancho y la altura del rectángulo medidos desde el centro hasta los bordes.

`ShowInfobox`
: Muestra el cuadro de información para el rectángulo si no está visible. De lo contrario, este método no tiene ningún efecto. Este método se puede usar para mostrar el cuadro de información incluso si EnableInfobox es falso.

Componentes del sensor - App Inventor para Android

Tabla de contenido

Sensor acelerómetro

Componente no visible que puede detectar sacudidas y medir la aceleración aproximadamente en tres dimensiones con unidades SI (m / s^2). Los componentes son:

- **xAccel** : 0 cuando el teléfono está en reposo sobre una superficie plana, positivo cuando el teléfono está inclinado hacia la derecha (es decir, su lado izquierdo está elevado) y negativo cuando el teléfono está inclinado hacia la izquierda (es decir, su tamaño correcto es elevado).

- **yAccel** : 0 cuando el teléfono está en reposo sobre una superficie plana, positivo cuando su parte inferior está elevada y negativo cuando su parte superior está elevada.
- **zAccel** : Igual a -9.8 (gravedad de la tierra en metros por segundo por segundo cuando el dispositivo está en reposo paralelo al suelo con la pantalla hacia arriba, 0 cuando es

perpendicular al suelo, y +9.8 cuando está hacia abajo. El valor también puede ser afectado al acelerarlo con o contra la gravedad.

Propiedades

Available
Enabled
LegacyMode (solo diseñador)
: Antes del lanzamiento que agregaba esta propiedad, el componente AccelerometerSensor pasaba directamente a través de los valores de los sensores recibidos del sistema Android. Sin embargo, estos valores no compensan las tabletas que cambian al modo Paisaje por defecto, lo que requiere que el programador MIT App Inventor lo compense. Sin embargo, la compensación daría lugar a resultados incorrectos en los dispositivos del modo Retrato, como los teléfonos. Ahora detectamos tabletas en modo Paisaje y realizamos la compensación. Sin embargo, si su proyecto ya está compensando el cambio, ahora obtendrá resultados incorrectos. Aunque nuestra solución preferida es que actualice su proyecto, también puede establecer esta propiedad como "verdadera" y nuestro código de compensación se desactivará. Nota:

MinimumInterval
: El intervalo mínimo, en milisegundos, entre los temblores del teléfono

Sensitivity
: Un número que codifica cuán sensible es el acelerómetro. Las opciones son: 1 = débil, 2 = moderada, 3 = fuerte.

XAccel
YAccel
ZAccel

Eventos

AccelerationChanged(number xAccel, number yAccel, number zAccel)
: Indica la aceleración cambiada en las dimensiones X, Y y Z.

Shaking()
: Indica que el dispositivo comenzó a agitarse o sigue agitándose.

Métodos

ninguna

Escáner de código de barras

Componente para usar el Barcode Scanner para leer un código de barras

Propiedades

Result
: Resultado del texto del escaneo anterior.

UseExternalScanner
: Si es cierto App Inventor buscará y usará un programa de escaneo externo como "Bar Code Scanner".

Eventos

AfterScan(text result)
: Indica que el escáner ha leído un resultado (de texto) y proporciona el resultado

Métodos

DoScan()
: Comienza un escaneo de código de barras, usando la cámara. Cuando finalice el escaneo, se generará el evento AfterScan.

Reloj

Componente no visible que proporciona el instante en el tiempo utilizando el reloj interno del teléfono. Puede disparar un temporizador a intervalos establecidos regularmente y realizar cálculos de tiempo, manipulaciones y conversiones.

Los métodos para convertir un instante al texto también están disponibles. Los patrones aceptables son cadenas vacías, MM / DD / AAAA HH: mm: ss a, o MMM d, aaaa HH: mm. La cadena vacía proporcionará el formato predeterminado, que es "MMM d, aaaa HH: mm: ss a" para FormatDateTime, "MMM d, aaaa" para FormatDate. Para ver todos los formatos posibles, mira aquí .

La fecha y la hora se formatean con InstantInTime y Duration.

- **Instantáneo** : consiste en año, mes, día de mes, hora, minuto y segundo. Se puede crear un instante utilizando el método MakeInstant.
- **Duración** : tiempo en milisegundos transcurrido entre instantes. La duración se puede obtener por el método de Duración.

Propiedades

`TimerAlwaysFires`
: Disparará incluso cuando la aplicación no se muestre en la pantalla si es verdadera

`TimerEnabled`
: Dispara el temporizador si es cierto

`TimerInterval`
: Intervalo entre eventos del temporizador en ms

Eventos

`Timer()`
: El temporizador se ha apagado.

Métodos

`InstantInTime AddDays(InstantInTime instant, number days)`
: Devuelve un instante en el tiempo algunos días después del argumento

`InstantInTime AddDuration(InstantInTime instant, Duration duration)`
: Devuelve un instante en el tiempo un tiempo después del argumento. La duración se puede obtener desde Clock.Duration ()

`InstantInTime AddHours(InstantInTime instant, number hours)`
: Devuelve un instante en el tiempo algunas horas después de la discusión

`InstantInTime AddMinutes(InstantInTime instant, number minutes)`
: Devuelve un instante en el tiempo algunos minutos después del argumento

`InstantInTime AddMonths(InstantInTime instant, number months)`
: Devuelve un instante en el tiempo algunos meses después de la discusión

`InstantInTime AddSeconds(InstantInTime instant, number seconds)`
: Devuelve un instante en el tiempo algunos segundos después del argumento

`InstantInTime AddWeeks(InstantInTime instant, number weeks)`
: Devuelve un instante en el tiempo algunas semanas después del argumento

`InstantInTime AddYears(InstantInTime instant, number years)`
: Devuelve un instante en el tiempo algunos años después de la discusión

`number DayOfMonth(InstantInTime instant)`
: Devuelve el día del mes (1-31) del instante

`Duration Duration(InstantInTime start, InstantInTime end)`
: Devuelve la duración, que es milisegundos transcurridos entre instantes

`number DurationToSeconds(Duration duration)`
: Convierte la duración a la cantidad de segundos.

`number DurationToMinutes(Duration duration)`
: Convierte la duración a la cantidad de minutos.

`number DurationToHours(Duration duration)`
: Convierte la duración en el número de horas.

`number DurationToDays(Duration duration)`
: Convierte la duración a la cantidad de días.

`number DurationToWeeks(Duration duration)`
: Convierte la duración a la cantidad de semanas.

`text FormatDate(InstantInTime instant, text pattern)`
: Devuelve texto que representa la fecha de un instante en el patrón especificado

`text FormatDateTime(InstantInTime instant, text pattern)`
: Devuelve texto que representa la fecha y la hora de un instante en el patrón especificado

`text FormatTime(InstantInTime instant)`
: Texto de retorno que representa el tiempo de un instante

`number GetMillis(InstantInTime instant)`
Devuelve el instante en el tiempo medido en milisegundos desde 1970.

`number Hour(InstantInTime instant)`
Devuelve la hora del día (0-23) del instante

`InstantInTime MakeInstant(text from)`
Devuelve un instante especificado por MM / DD / AAAA hh: mm: ss o MM / DD / AAAA o hh: mm. Un ejemplo de entrada de texto es "22/06/2015 12:18"

`InstantInTime MakeInstantFromMillis(number millis)`
Devuelve un instante en el tiempo especificado por milisegundos desde 1970.

`number Minute(InstantInTime instant)`
Devuelve el minuto de la hora (0-59) del instante

`number Month(InstantInTime instant)`
Devuelve el mes del año (1-12) del instante

`text MonthName(InstantInTime instant)`
Devuelve el nombre del mes desde el instante Ej. Enero, Febrero, Marzo ...

`InstantInTime Now()`
Devuelve un instante de la hora actual leída desde el reloj del teléfono

`number Second(InstantInTime instant)`
Devuelve el segundo minuto (0-59) del instante

`number SystemTime()`
Devuelve el tiempo interno del teléfono

`number Weekday(InstantInTime instant)`
Devuelve el día de la semana representado como un número de 1 (domingo) a 7 (sábado)

`text WeekdayName(InstantInTime instant)`
Devuelve el nombre del día de la semana del instante

`number Year(InstantInTime instant)`
Devuelve el año del instante

GyroscopeSensor

Componente no visible que puede medir la velocidad angular en tres dimensiones en unidades de grados por segundo.

Para funcionar, el componente debe tener su `Enabled`propiedad establecida en True, y el dispositivo debe tener un sensor de giroscopio.

Propiedades

`Available`
Indica si un sensor de giroscopio está disponible.

`Enabled`
Si está habilitado, entonces los eventos de sensor serán generados y `XAngularVelocity`, `YAngularVelocity`y `ZAngularVelocity`propiedades tendrán valores significativos.

`XAngularVelocity`
La velocidad angular alrededor del eje X, en grados por segundo.

`YAngularVelocity`
La velocidad angular alrededor del eje Y, en grados por segundo.

`ZAngularVelocity`
La velocidad angular alrededor del eje Z, en grados por segundo.

Eventos

`GyroscopeChanged(number xAngularVelocity, number yAngularVelocity, number zAngularVelocity, number timestamp)`
Indica que los datos del sensor del giroscopio han cambiado. El `timestamp`parámetro es el tiempo en nanosegundos en el que ocurrió el evento.

Métodos

ninguna

LocationSensor

Componente no visible que proporciona información de ubicación, incluida la longitud, la latitud, la altitud (si es compatible con el dispositivo), la velocidad (si es compatible con el dispositivo) y la dirección. Esto también puede realizar "geocodificación", convirtiendo una dirección dada (no necesariamente la actual) en una latitud (con el `LatitudeFromAddress`método) y una longitud (con el `LongitudeFromAddress`método).

Para funcionar, el componente debe tener su `Enabled`propiedad establecida en Verdadero, y el dispositivo debe tener detección de ubicación habilitada a través de redes inalámbricas o satélites GPS (si está al aire libre).
Es posible que la información de ubicación no esté disponible de inmediato cuando se inicia una aplicación. Tendrá que esperar un corto tiempo para encontrar y usar un proveedor de ubicación o esperar el evento OnLocationChanged.

Propiedades

`Accuracy`
`Altitude`
`AvailableProviders`
`CurrentAddress`
`DistanceInterval`
: Determina el intervalo de distancia mínima, en metros, que el sensor intentará utilizar para enviar actualizaciones de ubicación. Por ejemplo, si se establece en 5, el sensor disparará un evento LocationChanged solo después de que se hayan recorrido 5 metros. Sin embargo, el sensor no garantiza que se reciba una actualización exactamente en el intervalo de distancia. Puede llevar más de 5 metros disparar un evento, por ejemplo.

`Enabled`
`HasAccuracy`
`HasAltitude`
`HasLongitudeLatitude`
`Latitude`
`Longitude`
`ProviderLocked`
`ProviderName`
`TimeInterval`
: Determina el intervalo de tiempo mínimo, en milisegundos, que el sensor intentará utilizar para enviar actualizaciones de ubicación. Sin embargo, las actualizaciones de ubicación solo se recibirán cuando la ubicación del teléfono realmente cambie, y no se garantiza el uso del intervalo de tiempo especificado. Por ejemplo, si se usa 1000 como intervalo de tiempo, las actualizaciones de ubicación nunca se dispararán antes de 1000 ms, pero pueden dispararse en cualquier momento posterior.

Eventos

`LocationChanged(number latitude, number longitude, number altitude, number speed)`
: Indica que se ha detectado una nueva ubicación.

`StatusChanged(text provider, text status)`
: Indica que el estado del servicio del proveedor de ubicación ha cambiado, como cuando se pierde un proveedor o se empieza a utilizar un nuevo proveedor.

Métodos

`number LatitudeFromAddress(text locationName)`
: Deriva la latitud de la dirección dada

`number LongitudeFromAddress(text locationName)`
: Deriva la longitud de la dirección dada

Campo cercano

Componente no visible para proporcionar capacidades NFC. Por ahora, este componente solo admite la lectura y escritura de etiquetas de texto (si el dispositivo lo admite)

Para leer y escribir etiquetas de texto, el componente debe tener su `ReadMode`propiedad configurada en True o False, respectivamente.
Nota: Este componente solo funcionará en la Pantalla 1 de cualquier aplicación de App Inventor.

Propiedades

`LastMessage`
`ReadMode`
`TextToWrite`
`WriteType`

Eventos

`TagRead(text message)`
: Indica que se ha detectado una nueva etiqueta. Actualmente, esta es solo una etiqueta de texto sin formato, como se especifica en el manifiesto. Ver Compiler.java.

`TagWritten()`

Evento para TagWritten

Métodos

ninguna

Sensor de orientación

Use un componente de sensor de orientación para determinar la orientación espacial del teléfono.

Un sensor de orientación es un componente no visible que informa los siguientes tres valores, en grados:

- **Rollo** : 0 grados cuando el dispositivo está nivelado, aumentando a 90 grados a medida que el dispositivo se inclina hacia arriba sobre su lado izquierdo, y disminuyendo a -90 grados cuando el dispositivo se inclina hacia arriba sobre su lado derecho.

- **Inclinación** : 0 grados cuando el dispositivo está nivelado, aumentando a 90 grados a medida que el dispositivo se inclina, por lo que su parte superior apunta hacia abajo, luego disminuyendo a 0 grados a medida que se voltea. De manera similar, cuando el dispositivo está inclinado de manera que su parte inferior apunte hacia abajo, el tono disminuye a -90 grados, luego aumenta a 0 grados a medida que se vuelve hacia arriba.
- **Azimut** : 0 grados cuando la parte superior del dispositivo apunta hacia el norte, 90 grados cuando apunta hacia el este, 180 grados cuando apunta hacia el sur, 270 grados cuando apunta hacia el oeste, etc.

Estas medidas suponen que el dispositivo en sí no se está moviendo.

Propiedades

`Available`
: Indica si el sensor de orientación está presente en el dispositivo Android.

`Enabled`
: Si está configurado, el sensor de orientación está habilitado.

`Azimuth`
: Devuelve el ángulo acimutal del dispositivo.

`Pitch`
: Devuelve el ángulo de inclinación del dispositivo.

`Roll`
: Devuelve el ángulo de balanceo del dispositivo.

`Magnitude`
: Devuelve un número entre 0 y 1 que indica cuánto está inclinado el dispositivo. Da la magnitud de la fuerza que sentiría una bola rodando en la superficie del dispositivo.

`Angle`
: Devuelve un ángulo que indica la dirección en la que está mosaico el dispositivo. Es decir, indica la dirección de la fuerza que sentiría una bola rodando en la superficie del dispositivo.

Eventos

`OrientationChanged(number azimuth, number pitch, number roll)`
: Se llama cuando la orientación ha cambiado.

Podómetro

Un componente que actúa como un podómetro. Detecta movimiento a través del Accerleromter e intenta determinar si se ha realizado un paso. Usando una longitud de zancada configurable, también puede estimar la distancia recorrida.

Propiedades

`Distance`
: La distancia aproximada viajó en metros.

`ElapsedTime`

Tiempo transcurrido en milisegundos desde que se inició el podómetro.

SimpleSteps
La cantidad de pasos simples que se tomaron desde que comenzó el podómetro.

StopDetectionTimeout
La duración en milisegundos de inactividad (no se detectaron pasos) después de lo cual pasar a un estado "detenido"

StrideLength
Establece la longitud promedio de la zancada en metros.

WalkSteps
la cantidad de pasos a seguir desde que comenzó el podómetro.

Eventos

SimpleStep(number simpleSteps, number distance)
Este evento se ejecuta cuando se detecta un paso en bruto

WalkStep(number walkSteps, number distance)
Este evento se ejecuta cuando se detecta un paso caminando. Un paso de caminar es un paso que parece estar involucrado en el movimiento hacia adelante.

Métodos

Pause()
Pausa de conteo de pasos y distancia.

Reset()
Restablece el contador de pasos, la medida de distancia y el tiempo de ejecución.

Resume()
Continúa contando, sinónimo de inicio.

Save()
Guarda el estado del podómetro en el teléfono. Permisos permite la acumulación de pasos y la distancia entre las invocaciones de una aplicación que utiliza el podómetro. Las diferentes aplicaciones tendrán su propio estado guardado.

Start()
Comience a contar pasos

Stop()
Deja de contar pasos

Sensor de proximidad

Un componente del sensor que puede medir la proximidad de un objeto (en cm) con respecto a la pantalla de visualización de un dispositivo. Este sensor generalmente se utiliza para determinar si un teléfono está siendo llevado hasta el oído de una persona; es decir, le permite determinar qué tan lejos está un objeto de un dispositivo. Muchos dispositivos devuelven la distancia absoluta, en cm, pero algunos devuelven solo valores cercanos y lejanos. En este caso, el sensor generalmente informa su valor máximo de rango en el estado lejano y un valor menor en el estado cercano. Informa el siguiente valor:

- **Distancia** : la distancia desde el objeto al dispositivo

Propiedades

Available
Informa si el dispositivo tiene un sensor de proximidad

Enabled
Si está habilitado, el dispositivo escuchará los cambios en la proximidad

KeepRunningWhenOnPause
Si se establece en verdadero, seguirá detectando los cambios de proximidad incluso cuando la aplicación no sea visible

Distance
Devuelve la distancia desde el objeto al dispositivo

MaximumRange
Informa el rango máximo del ProximitySensor del dispositivo.

Eventos

ProximityChanged(number distance)
Se llama cuando cambia la distancia (en cm) del objeto al dispositivo.

Componentes sociales: App Inventor para Android

Tabla de contenido

ContactPicker

Un botón que, cuando se hace clic en, muestra una lista de los contactos para elegir. Después de que el usuario haya hecho una selección, las siguientes propiedades se establecerán en la información sobre el contacto elegido:

- `ContactName`: el nombre del contacto
- `EmailAddress`: la dirección de correo electrónico principal del contacto
- `EmailAddressList`: una lista de las direcciones de correo electrónico del contacto
- `ContactUri`: el URI del contacto en el dispositivo
- `PhoneNumber`: el número de teléfono principal del contacto (en versiones posteriores de Android)
- `PhoneNumberList`: una lista de los números de teléfono del contacto (en versiones posteriores de Android)
- `Picture`: el nombre del archivo que contiene la imagen del contacto, que se puede usar como un `Picture`valor de propiedad para el componente `Image`o `ImageSprite`.

Otras propiedades afectan a la apariencia del botón (`TextAlignment`, `BackgroundColor`, etc.) y si se puede hacer clic en (`Enabled`).

Es posible que el componente ContactPicker no funcione en todos los teléfonos. Por ejemplo, en los sistemas Android anteriores al sistema 3.0, no puede elegir números de teléfono, y la lista de direcciones de correo electrónico solo contendrá un correo electrónico.

Propiedades

`BackgroundColor`
: Devuelve el color de fondo del botón

`ContactName`
`EmailAddress`
`ContactUri`
: URI que especifica la ubicación del contacto en el dispositivo.

`Enabled`
`FontBold` (solo diseñador)
`FontItalic` (solo diseñador)
`FontSize` (solo diseñador)
`FontTypeface` (solo diseñador)
`Height`
`Image`
: Especifica la ruta de la imagen del botón. Si hay una Imagen y un Color de fondo, solo la Imagen estará visible.

`Picture`
`Shape` (solo diseñador)
: Especifica la forma del botón (predeterminado, redondeado, rectangular, oval). La forma no será visible si se muestra una imagen.

`ShowFeedback`
: Especifica si se debe mostrar un comentario visual para un botón que como una imagen como fondo.

`Text`
`TextAlignment` (solo diseñador)
`TextColor`
`Visible`

Especifica si el componente debe ser visible en la pantalla. El valor es verdadero si el componente se muestra y falso si está oculto.

`Width`

Eventos

`AfterPicking()`
El evento simple que se generará después de la actividad del selector devuelve su resultado y las propiedades se han completado.

`BeforePicking()`
Evento simple para plantear cuando se hace clic en el componente pero antes de que se inicie la actividad del selector.

`GotFocus()`
Indica que el cursor se movió sobre el botón, por lo que ahora es posible hacer clic en él.

`LostFocus()`
Indica que el cursor se alejó del botón, por lo que ya no es posible hacer clic en él.

Métodos

`Open()`
Abre el selector, como si el usuario hiciera clic en él.

`ViewContact(uri)`
Visualiza un contacto dado su URI.

EmailPicker

Un EmailPicker es un tipo de cuadro de texto. Si el usuario comienza a ingresar el nombre o la dirección de correo electrónico de un contacto, el teléfono mostrará un menú desplegable de opciones que completará la entrada. Si hay muchos contactos, el menú desplegable puede tardar varios segundos en aparecer, y puede mostrar resultados intermedios mientras se calculan las coincidencias.

Los contenidos iniciales del cuadro de texto y los contenidos <después de la entrada del usuario están en la `Text`propiedad. Si la `Text`propiedad está inicialmente vacía, el contenido de la `Hint`propiedad se mostrará débilmente en el cuadro de texto como una sugerencia para el usuario.
Otras propiedades afectan a la apariencia del cuadro de texto (`TextAlignment`, `BackgroundColor`, etc.) y si se puede utilizar (`Enabled`).
Los cuadros de texto como este se usan generalmente con `Button`componentes, con el usuario haciendo clic en el botón cuando se completa la entrada de texto.

Métodos

`RequestFocus()`
Establece el EmailPicker activo.

Propiedades

`BackgroundColor`
El color de fondo del cuadro de entrada. Puede elegir un color por nombre en el Diseñador o en el Editor de bloques. El color de fondo predeterminado es 'predeterminado' (aspecto tridimensional sombreado).

`Enabled`
Si el usuario puede ingresar texto en este cuadro de entrada. Por defecto, esto es cierto.

`FontBold` (solo diseñador)
Si la fuente para el texto debe ser negrita. Por defecto, no lo es.

`FontItalic` (solo diseñador)
Si el texto debe aparecer en cursiva. Por defecto, no.

`FontSize`
El tamaño de fuente para el texto. Por defecto, es 14.0 puntos.

`FontTypeface` (solo diseñador)
La fuente para el texto. El valor se puede cambiar en el Diseñador.

`Height`
`Hint`
Texto que debería aparecer débilmente en el cuadro de entrada para proporcionar una pista sobre lo que debe ingresar el usuario. Esto solo se puede ver si la `Text`propiedad está vacía.
`Text`

El texto en el cuadro de entrada, que puede ser configurado por el programador en Designer o Blocks Editor, o puede ser ingresado por el usuario (a menos que la `Enabled`propiedad sea falsa).

`TextAlignment` (solo diseñador)
Si el texto debe dejarse justificado, centrado o justificado a la derecha. Por defecto, el texto queda justificado.

`TextColor`
El color para el texto Puede elegir un color por nombre en el Diseñador o en el Editor de bloques. El color de texto predeterminado es negro.

`Visible`
Especifica si el componente debe ser visible en la pantalla. El valor es verdadero si el componente se muestra y falso si está oculto.

`Width`

Eventos

`GotFocus()`
Evento elevado cuando se selecciona este componente para la entrada, tal como por el usuario que lo toca.

`LostFocus()`
Evento generado cuando este componente ya no se selecciona para la entrada, como si el usuario tocara un cuadro de texto diferente.

Métodos

ninguna

Llamada telefónica

Un componente no visible que realiza una llamada telefónica al número especificado en la `PhoneNumber`propiedad, que se puede establecer en el Diseñador o en el Editor de bloques. El componente tiene un `MakePhoneCall`método que permite que el programa inicie una llamada telefónica.
A menudo, este componente se usa con el `ContactPicker`componente, que permite al usuario seleccionar un contacto de los almacenados en el teléfono y establece la `PhoneNumber`propiedad en el número de teléfono del contacto.
Para especificar directamente el número de teléfono (por ejemplo, 650-555-1212), establezca la `PhoneNumber`propiedad en un texto con los dígitos especificados (por ejemplo, "6505551212"). Se pueden incluir guiones, puntos y paréntesis (p. Ej., "(650) -555-1212") pero se ignorarán; espacios no pueden ser incluidos.

Propiedades

`PhoneNumber`

Eventos

`IncomingCallAnswered(text phoneNumber)`
Evento que indica que se responde una llamada telefónica entrante. phoneNumber es el número de teléfono de la llamada entrante.

`PhoneCallEnded(number status, text phoneNumber)`
Evento que indica que una llamada telefónica ha finalizado. Si el estado es 1, la llamada entrante se pierde o se rechaza; si el estado es 2, la llamada entrante se responde antes de colgar; si el estado es 3, la llamada saliente se cuelga. phoneNumber es el número de teléfono finalizado.

`PhoneCallStarted(number status, text phoneNumber)`
Evento que indica que una llamada telefónica ha comenzado. Si el estado es 1, la llamada entrante está sonando; si el estado es 2, se marca la llamada saliente. phoneNumber es el número de teléfono entrante / saliente.

Métodos

`MakePhoneCall()`
Realiza una llamada telefónica usando el número en la propiedad PhoneNumber.

PhoneNumberPicker

Un botón que, cuando se hace clic en, muestra una lista de los números de teléfono de los contactos para elegir. Después de que el usuario haya hecho una selección, las siguientes propiedades se establecerán en la información sobre el contacto elegido:

- `ContactName`: el nombre del contacto
- `PhoneNumber`: el número de teléfono del contacto
- `EmailAddress`: la dirección de correo electrónico del contacto
- `Picture`: el nombre del archivo que contiene la imagen del contacto, que se puede usar como un `Picture`valor de propiedad para el componente `Image`o `ImageSprite`.

Otras propiedades afectan a la apariencia del botón (`TextAlignment`, `BackgroundColor`, etc.) y si se puede hacer clic en (`Enabled`).

El componente PhoneNumberPicker puede no funcionar en todos los dispositivos Android. Por ejemplo, en sistemas Android anteriores al sistema 3.0, las listas devueltas de números de teléfono y direcciones de correo electrónico estarán vacías.

Propiedades

`BackgroundColor`
: Devuelve el color de fondo del botón

`ContactName`
`EmailAddress`
`Enabled`
`FontBold` (solo diseñador)
`FontItalic` (solo diseñador)
`FontSize` (solo diseñador)
`FontTypeface` (solo diseñador)
`Height`
`Image`
: Especifica la ruta de la imagen del botón. Si hay una Imagen y un Color de fondo, solo la Imagen estará visible.

`PhoneNumber`
`Picture`
`Shape` (solo diseñador)
: Especifica la forma del botón (predeterminado, redondeado, rectangular, oval). La forma no será visible si se muestra una imagen.

`ShowFeedback`
: Especifica si se debe mostrar un comentario visual para un botón que como una imagen como fondo.

`Text`
`TextAlignment` (solo diseñador)
`TextColor`
`Visible`
: Especifica si el componente debe ser visible en la pantalla. El valor es verdadero si el componente se muestra y falso si está oculto.

`Width`

Eventos

`AfterPicking()`
: El evento simple que se generará después de la actividad del selector devuelve su resultado y las propiedades se han completado.

`BeforePicking()`
: Evento simple para plantear cuando se hace clic en el componente pero antes de que se inicie la actividad del selector.

`GotFocus()`
: Indica que el cursor se movió sobre el botón, por lo que ahora es posible hacer clic en él.

`LostFocus()`
: Indica que el cursor se alejó del botón, por lo que ya no es posible hacer clic en él.

Métodos

`Open()`
: Abre el selector, como si el usuario hiciera clic en él.

Compartir

Compartir es un componente no visible que permite compartir archivos y / o mensajes entre su aplicación y otras aplicaciones instaladas en un dispositivo. El componente mostrará una lista de las aplicaciones instaladas que pueden manejar la información provista, y le permitirá al usuario elegir una para compartir el contenido, por ejemplo, una aplicación de correo, una aplicación de red social, una aplicación de mensajes de texto, y más.
La ruta del archivo se puede tomar directamente de otros componentes, como la Cámara o el ImagePicker, pero también se puede especificar directamente para leer desde el almacenamiento. Tenga en cuenta que los diferentes dispositivos tratan el almacenamiento de manera diferente, por lo tanto, algunas cosas que debe probar si, por ejemplo, tiene un archivo llamado arrow.gif en la carpeta `Appinventor/assets`, serían:

- `"file:///sdcard/Appinventor/assets/arrow.gif"`

o

- `"/storage/Appinventor/assets/arrow.gif"`

Propiedades

ninguna

Eventos

ninguna

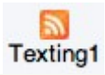
Métodos

`ShareFile(text file)`
: Comparte un archivo a través de cualquier aplicación capaz instalada en el teléfono mostrando una lista de las aplicaciones disponibles y permitiendo que el usuario elija una de la lista. La aplicación seleccionada se abrirá con el archivo insertado en ella.

`ShareFileWithMessage(text file, text message)`
: Comparte un archivo y un mensaje a través de cualquier aplicación capaz instalada en el teléfono mostrando una lista de aplicaciones disponibles y permitiendo al usuario elegir una de la lista. La aplicación seleccionada se abrirá con el archivo y el mensaje insertados en ella.

`ShareMessage(text message)`
: Comparte un mensaje a través de cualquier aplicación capaz instalada en el teléfono mostrando una lista de las aplicaciones disponibles y permitiendo al usuario elegir una de la lista. La aplicación seleccionada se abrirá con el mensaje insertado en ella.

Enviar mensajes de texto



Un componente que, cuando `SendMessage` se llama al método, envía el mensaje de texto especificado en la `Message` propiedad al número de teléfono especificado en la`PhoneNumber` propiedad.
Si la `ReceivingEnabled` propiedad está configurada en 1, los mensajes **no** se recibirán. Si `ReceivingEnabled` se establece en 2, los mensajes se recibirán solo cuando la aplicación se esté ejecutando. Finalmente, si `ReceivingEnabled` se establece en 3, se recibirán mensajes cuando la aplicación se esté ejecutando **y** cuando la aplicación no se esté ejecutando, se pondrán en cola y se mostrará una notificación al usuario.
Cuando llega un mensaje, el `MessageReceived` evento se levanta y proporciona el número de envío y el mensaje.

Una aplicación que incluye este componente recibirá mensajes incluso cuando esté en segundo plano (es decir, cuando no esté visible en la pantalla) y, lo que es más, incluso si la aplicación no se está ejecutando, siempre que esté instalada en el teléfono. Si el teléfono recibe un mensaje de texto cuando la aplicación no está en primer plano, el teléfono mostrará una notificación en la barra de notificaciones. Al seleccionar la notificación aparecerá la aplicación. Como desarrollador de aplicaciones, es probable que desee otorgar a sus usuarios la posibilidad de controlar ReceivingEnabled para que el teléfono pueda ignorar los mensajes de texto.

Si la propiedad GoogleVoiceEnabled es verdadera, los mensajes se pueden enviar a través de WiFi usando Google Voice. Esta opción requiere que el usuario tenga una cuenta de Google Voice y que la aplicación de voz móvil esté instalada en el teléfono. La opción Google Voice solo funciona en teléfonos compatibles con Android 2.0 (Eclair) o superior.

Para especificar el número de teléfono (por ejemplo, 650-555-1212), establezca la `PhoneNumber` propiedad en una cadena de texto con los dígitos especificados (por ejemplo, 6505551212). Se pueden incluir guiones, puntos y paréntesis (p. Ej., (650) -555-1212) pero se ignorarán; espacios no pueden ser incluidos.

Otra forma de que una aplicación especifique un número de teléfono sería incluir un `PhoneNumberPicker` componente que permita a los usuarios seleccionar un número de teléfono de los que están almacenados en los contactos del teléfono.

Propiedades

`GoogleVoiceEnabled`
: Si es verdadero, SendMessage intentará enviar mensajes a través de WiFi utilizando Google Voice. Esto requiere que la aplicación Google Voice debe estar instalada y configurada en el teléfono o tableta, con una cuenta de Google Voice. Si GoogleVoiceEnabled es falso, el dispositivo debe tener un servicio de teléfono y mensajes de texto para enviar o recibir mensajes con este componente.

`Message`
: El mensaje que se enviará cuando se llame al método SendMessage.

`PhoneNumber`
: El número al que se enviará el mensaje cuando se llame al método SendMessage. El número es una cadena de texto con los dígitos especificados (por ejemplo, 6505551212). Se pueden incluir guiones, puntos y paréntesis (p. Ej., (650) -555-1212) pero se ignorarán; los espacios no deben ser incluidos.

`ReceivingEnabled`
: Si se establece en 1 (OFF), no se recibirán mensajes. Si se establece en 2 (FOREGROUND) o3 (SIEMPRE), el componente responderá a los mensajes si se está ejecutando. Si theapp no se está ejecutando, el mensaje se descartará si se establece en 2 (FOREGROUND). Si se establece en 3 (SIEMPRE) y la aplicación no se está ejecutando, el teléfono mostrará una notificación. Al seleccionar la notificación aparecerá la señal del dispositivo, el evento MessageReceived. Los mensajes recibidos cuando la aplicación está inactiva se pondrán en cola, por lo que varios eventos MessageReceived pueden aparecer cuando la aplicación se active. Como desarrollador de aplicaciones,

Eventos

`MessageReceived(text number, text messageText)`
: Evento que se produce cuando el teléfono recibe un mensaje de texto.

Métodos

`SendMessage()`
: Enviar un mensaje de texto

Gorjeo

Un componente no visible que permite la comunicación con Twitter . Una vez que un usuario ha iniciado sesión en su cuenta de Twitter (y la autorización ha sido confirmada como exitosa por el `IsAuthorized`evento), hay muchas más operaciones disponibles:

- Buscando Twitter para tweets o etiquetas (`SearchTwitter`)
- Enviando un Tweet (`Tweet`)
- Enviar un Tweet con una Imagen (`TweetWithImage`)
- Dirigiendo un mensaje a un usuario específico (`DirectMessage`)
- Recibir los mensajes más recientes dirigidos al usuario que ha iniciado sesión (`RequestDirectMessages`)
- Siguiendo a un usuario específico (`Follow`)
- Dejar de seguir a un usuario específico (`StopFollowing`)
- Obtener una lista de usuarios que siguen al usuario conectado (`RequestFollowers`)
- Obtener los mensajes más recientes de los usuarios seguidos por el usuario que ha iniciado sesión (`RequestFriendTimeline`)
- Obtener las menciones más recientes del usuario conectado (`RequestMentions`)

Debe obtener una clave del consumidor y un secreto del consumidor para la autorización de Twitter específica para su aplicación en http://twitter.com/oauth_clients/new

Propiedades

`ConsumerKey`
: La clave del consumidor que se utilizará al autorizar con Twitter a través de OAuth.

`ConsumerSecret`
: El secreto del consumidor que se utilizará al autorizar con Twitter a través de OAuth

`DirectMessages`
: Esta propiedad contiene una lista de los mensajes más recientes que mencionan al usuario que inició sesión. Inicialmente, la lista está vacía. Para configurarlo, el programa debe:

    1. Llamar al `Authorize`método

2. Espera el `Authorized`evento.
3. Llamar al `RequestDirectMessages`método
4. Espera el `DirectMessagesReceived`evento.

El valor de esta propiedad se establecerá en la lista de mensajes directos recuperados (y mantendrá ese valor hasta cualquier llamada posterior a `RequestDirectMessages`).

`Followers`

Esta propiedad contiene una lista de los seguidores del usuario que ha iniciado sesión. Inicialmente, la lista está vacía. Para configurarlo, el programa debe:

1. Llamar al `Authorize`método
2. Espera el `IsAuthorized`evento.
3. Llamar al `RequestFollowers`método
4. Espera el `FollowersReceived`evento.

El valor de esta propiedad se establecerá en la lista de seguidores (y mantendrá su valor hasta cualquier llamada posterior a `RequestFollowers`).

`FriendTimeline`

Esta propiedad contiene los 20 mensajes más recientes de usuarios que se están siguiendo. Inicialmente, la lista está vacía. Para configurarlo, el programa debe:

1. Llamar al `Authorize`método
2. Espera el `IsAuthorized`evento.
3. Especifique los usuarios a seguir con una o más llamadas al `Follow`método.
4. Llamar al `RequestFriendTimeline`método
5. Espera el `FriendTimelineReceived`evento.

El valor de esta propiedad se establecerá en la lista de mensajes (y mantendrá su valor hasta cualquier llamada posterior a `RequestFriendTimeline`.

`Mentions`

Esta propiedad contiene una lista de menciones del usuario que ha iniciado sesión. Inicialmente, la lista está vacía. Para configurarlo, el programa debe:

1. Llamar al `Authorize`método
2. Espera el `IsAuthorized`evento.
3. Llamar al `RequestMentions`método
4. Espera el `MentionsReceived`evento.

El valor de esta propiedad se establecerá en la lista de menciones (y mantendrá su valor hasta cualquier llamada posterior a `RequestMentions`).

`SearchResults`

Esta propiedad, que inicialmente está vacía, se establece en una lista de resultados de búsqueda después del programa:

1. Llama al `SearchTwitter`método.
2. Espera el `SearchSuccessful`evento.

El valor de la propiedad será el mismo que el parámetro para `SearchSuccessful`. Tenga en cuenta que no es necesario llamar al `Authorize`método antes de llamar `SearchTwitter`.

`Username`

El nombre de usuario del usuario autorizado. Vacío si no hay un usuario autorizado.

Eventos

`DirectMessagesReceived(list messages)`

Este evento se produce cuando `RequestDirectMessages`se han recuperado los mensajes recientes solicitados . Una lista de los mensajes se puede encontrar en el `messages`parámetro o la `Messages`propiedad.

`FollowersReceived(list followers2)`

Este evento se produce cuando `RequestFollowers`se han recuperado todos los seguidores del usuario conectado solicitado . Una lista de los seguidores se puede encontrar en el `followers`parámetro o la `Followers`propiedad.

`FriendTimelineReceived(list timeline)`

Este evento se produce cuando los mensajes solicitados `RequestFriendTimeline`han sido recuperados. El `timeline`parámetro y la `Timeline`propiedad contendrán una lista de listas, donde cada sub-lista contiene una actualización de estado del formulario (mensaje de nombre de usuario)

`IsAuthorized()`

Este evento se produce después de que el programa llama `Authorize`si la autorización fue exitosa. También se llama después de una llamada a `CheckAuthorized`si ya tenemos un token de acceso válido. Después de que se haya elevado este evento, se puede llamar a cualquier otro método para este componente.

`MentionsReceived(list mentions)`

Este evento se produce cuando `RequestMentions`se recuperan las menciones del usuario conectado solicitado . Una lista de las menciones se puede encontrar en el `mentions`parámetro o la `Mentions`propiedad.

`SearchSuccessful(list searchResults)`

Este evento se genera cuando `SearchSuccessful`se recuperan los resultados de la búsqueda solicitada . Una lista de los resultados se puede encontrar en el `results`parámetro o la `Results`propiedad.

Métodos

`Authorize()`

Redirige al usuario para iniciar sesión en Twitter a través del navegador web utilizando el protocolo OAuth si aún no tenemos autorización.

`CheckAuthorized()`

Comprueba si ya tenemos acceso y, de ser así, hace que se llame al controlador de eventos IsAuthorized.

`DeAuthorize()`

Elimina la autorización de Twitter de esta instancia de aplicación en ejecución

`DirectMessage(text user, text message)`

Esto envía un mensaje directo (privado) al usuario especificado. El mensaje se recortará si supera los 160 caracteres.

Requisitos : esto solo debe invocarse una vez `IsAuthorized`que se ha generado el evento, lo que indica que el usuario ha iniciado sesión correctamente en Twitter.

`Follow(text user)`

Comienza siguiendo a un usuario.

`RequestDirectMessages()`

Solicita los 20 mensajes directos más recientes enviados al usuario que ha iniciado sesión. Cuando los mensajes han sido recuperados, el sistema levantará el `DirectMessagesReceived`evento y establecerá la `DirectMessages`propiedad en la lista de mensajes.

Requisitos : esto solo debe invocarse una vez `IsAuthorized`que se ha generado el evento, lo que indica que el usuario ha iniciado sesión correctamente en Twitter.

`RequestFollowers()`

Obtiene quién te está siguiendo.

`RequestFriendTimeline()`

Obtiene los 20 mensajes más recientes en la línea de tiempo del usuario.

`RequestMentions()`

Solicita las 20 menciones más recientes del usuario que ha iniciado sesión. Cuando se hayan recuperado las menciones, el sistema levantará el `MentionsReceived`evento y establecerá la `Mentions`propiedad en la lista de menciones.

Requisitos : esto solo debe invocarse una vez `IsAuthorized`que se ha generado el evento, lo que indica que el usuario ha iniciado sesión correctamente en Twitter.

`SearchTwitter(text query)`

Esto busca en Twitter la consulta de Cadena dada.

Requisitos : esto solo debe invocarse una vez `IsAuthorized`que se ha generado el evento, lo que indica que el usuario ha iniciado sesión correctamente en Twitter.

`StopFollowing(text user)`

Detiene a un usuario.

`Tweet(text status)`

Esto envía un tweet como usuario registrado con el texto especificado, que se recortará si supera los 160 caracteres.

Requisitos : esto solo debe invocarse una vez `IsAuthorized`que se ha generado el evento, lo que indica que el usuario ha iniciado sesión correctamente en Twitter.

`TweetWithImage(text status, text imagePath)`

Esto envía un tweet como usuario registrado con el texto especificado y una ruta a la imagen que se cargará, que se recortará si supera los 160 caracteres. Si no se encuentra una imagen o no es válida, la actualización no se enviará.

Requisitos : esto solo debe invocarse una vez `IsAuthorized`que se ha generado el evento, lo que indica que el usuario ha iniciado sesión correctamente en Twitter.

Almacenamiento: App Inventor para Android

Tabla de contenido

Archivo

Componente no visible para almacenar y recuperar archivos. Use este componente para escribir o leer archivos en su dispositivo. El comportamiento predeterminado es escribir archivos en el directorio de datos privados asociado con su aplicación. El acompañante escribe archivos en / sdcard / AppInventor / data para una fácil depuración. Si la ruta del archivo comienza con una barra inclinada (/), entonces el archivo se crea con relación a / sdcard. Por ejemplo, escribir un archivo en /myFile.txt escribirá el archivo en /sdcard/myFile.txt.

Propiedades

ninguna

Eventos

`AfterFileSaved(text fileName)`
: Evento que indica que el contenido del archivo se ha escrito.

`GotText(text text)`
: Evento que indica que los contenidos del archivo han sido leídos.

Métodos

`AppendToFile(text text, text fileName)`
: Agrega texto al final de un archivo. Crea el archivo si aún no existe. Consulte el texto de ayuda en SaveFile para obtener información sobre dónde se escriben los archivos.

`Delete(text fileName)`
: Elimina un archivo del almacenamiento. Prefija el nombre de archivo con / para eliminar un archivo específico en la tarjeta SD (por ejemplo, /myFile.txt eliminará el archivo /sdcard/myFile.txt). Si el nombre de archivo no comienza con un /, se eliminará el archivo ubicado en el almacenamiento privado del programa. Iniciar el archivo con // es un error porque los archivos de activos no se pueden eliminar.

`ReadFrom(text fileName)`
: Lee texto de un archivo en el almacenamiento. Prefija el nombre de archivo con / para leer de un archivo específico en la tarjeta SD (por ejemplo, /myFile.txt leerá el archivo /sdcard/myFile.txt). Para leer los activos empaquetados con una aplicación (también funciona para el Companion), inicie el nombre de archivo con // (dos barras inclinadas). Si un nombre de archivo no comienza con una barra inclinada, se leerá desde el almacenamiento privado de la aplicación (para aplicaciones empaquetadas) y desde / sdcard / AppInventor / data para el Companion.

`SaveFile(text text, text fileName)`
: Guarda texto en un archivo. Si el nombre del archivo comienza con una barra inclinada (/), el archivo se escribe en la tarjeta SD (por ejemplo, escribiendo en / myFile.txt escribirá el archivo en /sdcard/myFile.txt). Si el nombre de archivo no comienza con una barra inclinada, se escribirá en el directorio de datos privados del programa, donde no será accesible para otros programas en el teléfono. Hay una excepción especial para AI Companion donde estos archivos se escriben en / sdcard / AppInventor / data para facilitar la depuración. Tenga en cuenta que este bloque sobrescribirá un archivo si ya existe. Si desea agregar contenido a un archivo, use el bloque de agregar.

FusionTablesControl

Un componente no visible que se comunica con Google Fusion Tables. Fusion Tables le permite almacenar, compartir, consultar y visualizar tablas de datos; este componente le permite consultar, crear y modificar estas tablas.

Este componente usa la API de Fusion Tables V2.0 .

Las aplicaciones que usan Fusion Tables deben autenticarse con los servidores de Google. Hay dos formas de hacerlo. La primera forma solo utiliza una clave API que usted (el desarrollador) obtiene. Con este enfoque, los usuarios finales también deben iniciar sesión para acceder a una tabla Fusion.

El segundo enfoque es usar la Autenticación de servicio. Con este enfoque, creas credenciales y una "Dirección de correo electrónico de cuenta de servicio" especial que permite a los usuarios finales editar tus tablas Fusion sin iniciar sesión; su cuenta de servicio autentica todo acceso.

Uso del componente FusiontablesControl

Creando Fusion Tables

Es probable que desee crear sus propias tablas Fusion para experimentar mientras desarrolla sus aplicaciones. Esto es tan fácil como crear un documento de Google, si está familiarizado con ese proceso. Estos son los pasos:

1. En la web, inicie sesión en su cuenta de Gmail o en cualquier otro servicio de Google (p. Ej., Drive, YouTube). Navega a Google Drive.
2. Haga clic en el botón *Nuevo* y navegue a *Más* . Si no ve una opción de Google Fusion Tables, seleccione *Conectar más aplicaciones* y desplácese por la página de servicios de Google para buscar el servicio *Fusion Tables* y conectarlo a su Google Drive.
3. Es posible que desee ver algunos de los ejemplos y trabajar a través de un tutorial (por ejemplo, Pizza Party Tutorial) para aprender los conceptos básicos.
4. Haga clic en el botón *Ver mis tablas* (arriba a la derecha de la página). Esto te llevará a tu propia página.
5. Debería ver una lista de sus propias tablas o tablas que haya compartido con usted (posiblemente ninguna).
6. Use el botón *Crear* para crear una nueva tabla. Dale algunos nombres de columna y guárdalo.
7. Haga clic en el botón *Compartir* (arriba a la derecha) para modificar los permisos de la tabla.

Creando una aplicación Fusiontables

Cuando arrastre el componente *FusiontablesControl* al Diseñador, no olvide configurar su propiedad *ApiKey* , que inicialmente está en blanco. Debe copiar esto desde su Google Developers Console y pegarlo en el campo de propiedad.
Para obtener una **clave API** , siga estas instrucciones.

1. Vaya a su Google Developers Console e inicie sesión si es necesario.
2. En *APIs y autenticación,* seleccione el elemento de *API* en el menú de la izquierda.
3. Elija la *API* de *Fusion Tables* de la lista proporcionada y actívela.
4. En la barra izquierda, selecciona el elemento de *Credenciales* .
5. Debajo del *acceso a la API pública,* haga clic en *Crear nueva clave* , elija la *clave de Android* y haga clic en *Crear* para generar una clave de API.

Su (s) clave (s) de API aparecerán en el panel al lado de "Acceso público a API". Debe proporcionar esa clave como el valor de la propiedad *ApiKey* en su aplicación Fusion Tables.
Una vez que tenga una clave API, establezca el valor de la propiedad *Query* en una consulta SQL Fusiontables válida y llame a *SendQuery* para ejecutar la consulta. App Inventor enviará la consulta al servidor de Fusion Tables y el bloque *GotResult* se activará cuando se devuelva un resultado del servidor. Los resultados de la consulta se devolverán en formato CSV y se pueden convertir a formato de lista utilizando los bloques "lista de la tabla csv" o "lista de la fila csv".
Tenga en cuenta que no necesita preocuparse por la codificación UTF de la consulta. Pero debe asegurarse de que la consulta sigue la sintaxis descrita en el manual de referencia , lo que significa que cosas como mayúsculas para los nombres de las columnas son importantes y que las comillas simples deben usarse alrededor de los nombres de las columnas si hay espacios en ellas.
Para configurar la **Autenticación del servicio** para su Fusion Table, siga estos pasos adicionales:

1. En la API de Google, la consola en *APIs y auth* selecciona el elemento *API* en el menú de la izquierda.
2. Haga clic en el botón *Crear nueva ID de cliente* . Seleccione la opción *Cuenta de servicio* y haga clic en *Crear ID de cliente* .
3. Un archivo llamado *KeyFile* (termina en la extensión .p12) se descargará automáticamente en su computadora. Guárdalo en un lugar que recordarás. Una vez que la creación esté completa, obtendrá una tabla con la información de su Cuenta de servicio.
4. En la ventana del diseñador de App Inventor, seleccione FusionTablesControl. En el panel de propiedades, agregue *ServiceAccountEmail* (de la tabla en la consola), cargue KeyFile y marque la casilla *UseServiceAuthentication* .
5. Comparta Fusion Table con su ServiceAccountEmail y asígnele permisos de edición, del mismo modo que compartiría cualquier otro Google Doc con una dirección de correo electrónico.

Propiedades

`ApiKey`
: Tu clave de Google API. Consulte arriba para obtener detalles sobre cómo obtener una clave API.

`KeyFile`

Especifica la ruta del archivo de clave privada. Este archivo de clave se usa para obtener acceso a la API de FusionTables a través de la Autenticación de servicio.

`Query`
La consulta para enviar a la API de Fusion Tables.

Para ver ejemplos y formatos de consultas legales, consulte el manual de referencia de Fusion Tables API v2.0 .

Tenga en cuenta que no necesita preocuparse por la codificación UTF de la consulta. Debe asegurarse de que la consulta siga la sintaxis descrita en el manual de referencia. Tenga en cuenta que las mayúsculas para los nombres de las columnas son necesarias y que las comillas simples se deben usar alrededor de los nombres de las columnas si hay espacios en ellas.

`ServiceAccountEmail`
La dirección de correo electrónico de la cuenta de servicio utilizada para la autenticación del servicio.

`UseServiceAuthentication`
Indica si se debe usar una cuenta de servicio para la autenticación.

Eventos

`GotResult(text result)`
Indica que la consulta de Fusion Tables ha finalizado el procesamiento y se ha devuelto con un resultado. El resultado de la consulta generalmente se devolverá en formato CSV y se puede convertir a formato de lista utilizando los bloques "lista de la tabla csv" o "lista de la fila csv".

Métodos

`DoQuery()`
OBSOLETO. Este bloque está en desuso a fines de 2012. Use SendQuery en su lugar.

`ForgetLogin()`
Olvide las credenciales de inicio de sesión del usuario final. No tiene ningún efecto en la Autenticación del servicio.

`GetRows(text tableId, text columns)`
Obtiene todas las filas de una tabla Fusion especificada. El campo tableId (obligatorio) es el id de Fusion Table. El campo de columnas es una lista de columnas separadas por comas para recuperar.

`GetRowsWithConditions(text tableId, text columns, text conditions)`
Obtiene todas las filas de una tabla Fusion que cumplen ciertas condiciones. El campo tableId (obligatorio) es el id de Fusion Table. El campo de columnas es una lista de columnas separadas por comas para recuperar. El campo de condiciones especifica qué filas recuperar de la tabla (por ejemplo, las filas en las que un valor de columna particular no es nulo).

`InsertRow(text tableId, text columns, text values)`
Inserta una fila en la tabla Fusion especificada. El campo tableId es el id de Fusion Table. El campo de columnas es una lista separada por comas de las columnas en las que insertar valores. El campo de valores especifica qué valores insertar en cada columna.

`SendQuery()`
Envíe la consulta al servidor de Fusion Tables.

TinyDB

TinyDB es un componente no visible que almacena datos para una aplicación.

Las aplicaciones creadas con App Inventor se inicializan cada vez que se ejecutan. Esto significa que si una aplicación establece el valor de una variable y el usuario abandona la aplicación, el valor de esa variable no se recordará la próxima vez que se ejecute la aplicación. Por el contrario, TinyDB es un almacén de datos *persistente* para la aplicación. Los datos almacenados en un TinyDB estarán disponibles cada vez que se ejecute la aplicación. Un ejemplo podría ser un juego que guarda la puntuación más alta y la recupera cada vez que se juega.
Los elementos de datos son cadenas almacenadas bajo *etiquetas* . Para almacenar un elemento de datos, especifique la etiqueta en la que debe estar almacenado. Posteriormente, puede recuperar los datos que se almacenaron bajo una etiqueta determinada.
Cada aplicación tiene su propio almacén de datos. Solo hay una tienda de datos por aplicación. Incluso si tiene múltiples componentes TinyDB, usarán el mismo almacén de datos. Para obtener el efecto de tiendas separadas, use diferentes claves. No puede usar TinyDB para pasar datos entre dos aplicaciones diferentes en el teléfono, aunque *puede* usar TinyDB para compartir datos entre las diferentes pantallas de una aplicación de pantallas múltiples.

Cuando desarrolles aplicaciones usando AI Companion, todas las aplicaciones que usen ese Companion compartirán el mismo TinyDB. Ese intercambio desaparecerá una vez que las

aplicaciones se hayan empaquetado e instalado en el teléfono. Durante el desarrollo, debe tener cuidado de borrar los datos de la aplicación Companion cada vez que comience a trabajar en una nueva aplicación.

Propiedades

ninguna

Eventos

ninguna

Métodos

`ClearAll()`
: Borre todo el almacén de datos en TinyDB.

`ClearTag(text tag)`
: Borre la entrada con la etiqueta dada.

`any GetTags()`
: Devuelve una lista de todas las etiquetas en TinyDB.

`any GetValue(text tag, any valueIfTagNotThere)`
: Recupere el valor almacenado bajo la etiqueta dada. Si no existe dicha etiqueta, devuelve valueIfTagNotThere.

`StoreValue(text tag, any valueToStore)`
: Almacene el valor debajo de la etiqueta dada. El almacenamiento persiste en el teléfono cuando se reinicia la aplicación.

TinyWebDB

Componente no visible que se comunica con un servicio web para almacenar y recuperar información.

Consulte Creación de un servicio TinyWebDB personalizado.

Propiedades

`ServiceURL`
: La URL de la base de datos con la que el componente debe comunicarse.

Eventos

`GotValue(text tagFromWebDB, any valueFromWebDB)`
: Indica que una solicitud del servidor GetValue ha tenido éxito.

`ValueStored()`
: Evento que indica que una solicitud del servidor StoreValue ha tenido éxito.

`WebServiceError(text message)`
: Indica que la comunicación con el servicio web señaló un error.

Métodos

`GetValue(text tag)`
: Envía una solicitud al servicio web para obtener el valor almacenado en la etiqueta especificada. El servicio web debe decidir qué devolver si no hay ningún valor almacenado debajo de la etiqueta. Este componente acepta lo que se devuelve.

`StoreValue(text tag, any valueToStore)`
: Envía una solicitud al servicio web para almacenar el valor dado bajo la etiqueta dada.

Componentes de conectividad: App Inventor para Android

Tabla de contenido

- ActivityStarter
- BluetoothClient
- BluetoothServer
- Web

ActivityStarter

Un componente que puede iniciar una actividad usando el `StartActivity`método.

Las actividades que se pueden lanzar incluyen:

- Iniciando otra App Inventor para la aplicación de Android. Para hacerlo, primero encuentre la *clase* de la otra aplicación descargando el código fuente y usando un explorador de archivos o descomprima la utilidad para encontrar un archivo llamado "youngandroidproject / project.properties". La primera línea del archivo comenzará con "main =" y será seguido por el nombre de la clase; por ejemplo `main=com.gmail.Bitdiddle.Ben.HelloPurr.Screen1`,. (Los primeros componentes indican que fue creado por Ben.Bitdiddle@gmail.com). Para que `ActivityStarter`lance esta aplicación, establezca las siguientes propiedades:
    - `ActivityPackage`al nombre de la clase, descartando el último componente (por ejemplo, `com.gmail.Bitdiddle.Ben.HelloPurr`)
    - `ActivityClass`a todo el nombre de clase (por ejemplo, `com.gmail.Bitdiddle.Ben.HelloPurr.Screen1`)
- Iniciando la aplicación de la cámara estableciendo las siguientes propiedades:
    - `Action: android.intent.action.MAIN`
    - `ActivityPackage: com.android.camera`
    - `ActivityClass: com.android.camera.Camera`
- Realizando búsqueda web. Suponiendo que el término que desea buscar es "vampiro" (siéntase libre de sustituirlo por su elección), establezca las propiedades en:
    - `Action: android.intent.action.WEB_SEARCH`
    - `ExtraKey: query`
    - `ExtraValue: vampire`
    - `ActivityPackage: com.google.android.providers.enhancedgooglesearch`
    - `ActivityClass: com.google.android.providers.enhancedgooglesearch.Launcher`
- Abrir un navegador a una página web específica. Suponiendo que la página a la que desea ir es "www.facebook.com" (siéntase libre de sustituir su elección), establezca las propiedades en:
    - `Action: android.intent.action.VIEW`
    - `DataUri: http://www.facebook.com`

Propiedades

`Action`
`ActivityClass`
`ActivityPackage`
`DataType`
`DataUri`
`Extras`
: Acepta una lista de pares que se utilizan como pares clave / valor en el campo "Extra" de la actividad

`ExtraKey`*`(Obsolete)`*
`ExtraValue`*`(Obsolete)`*
`Result`
`ResultName`
`ResultType`
`ResultUri`

Eventos

`AfterActivity(text result)`
: Evento generado después de que este ActivityStarter regrese.

`ActivityCanceled()`
: Evento generado si este ActivityStarter regresa porque la actividad fue cancelada.

Métodos

`text ResolveActivity()`
: Devuelve el nombre de la actividad que corresponde a este ActivityStarter o una cadena vacía si no se puede encontrar la actividad correspondiente.

`StartActivity()`
: Inicie la actividad correspondiente a este ActivityStarter.

BluetoothClient

Componente del cliente Bluetooth

Propiedades

AddressesAndNames
: Las direcciones y los nombres de los dispositivos Bluetooth emparejados

Available
: Si Bluetooth está disponible en el dispositivo

CharacterEncoding

DelimiterByte

Enabled
: Si Bluetooth está habilitado

HighByteFirst

IsConnected

Secure
: Ya sea para invocar SSP (Simple Secure Pairing), que es compatible con dispositivos con Bluetooth v2.1 o superior. Al trabajar con dispositivos Bluetooth incrustados, es posible que esta propiedad deba establecerse en False. Para Android 2.0-2.2, esta configuración de propiedad será ignorada.

Eventos

ninguna

Métodos

number BytesAvailableToReceive()
: Devuelve una estimación de la cantidad de bytes que se pueden recibir sin bloquear

boolean Connect(text address)
: Conéctese al dispositivo Bluetooth con la dirección especificada y el perfil de puerto serie (SPP). Devuelve verdadero si la conexión fue exitosa.

boolean ConnectWithUUID(text address, text uuid)
: Conéctese al dispositivo Bluetooth con la dirección especificada y el UUID. Devuelve verdadero si la conexión fue exitosa.

Disconnect()
: Desconectarse del dispositivo Bluetooth conectado.

boolean IsDevicePaired(text address)
: Comprueba si el dispositivo Bluetooth con la dirección especificada está emparejado.

number ReceiveSigned1ByteNumber()
: Reciba un número de 1 byte firmado desde el dispositivo Bluetooth conectado.

number ReceiveSigned2ByteNumber()
: Reciba un número de 2 bytes firmado desde el dispositivo Bluetooth conectado.

number ReceiveSigned4ByteNumber()
: Recibe un número de 4 bytes firmado del dispositivo Bluetooth conectado.

list ReceiveSignedBytes(number numberOfBytes)
: Reciba múltiples valores de bytes firmados desde el dispositivo Bluetooth conectado. Si numberOfBytes es menor que 0, lea hasta que se reciba un valor de byte delimitador.

text ReceiveText(number numberOfBytes)
: Recibe texto del dispositivo Bluetooth conectado. Si numberOfBytes es menor que 0, lea hasta que se reciba un valor de byte delimitador.

number ReceiveUnsigned1ByteNumber()
: Reciba un número de 1 byte sin firmar del dispositivo Bluetooth conectado.

number ReceiveUnsigned2ByteNumber()
: Reciba un número de 2 bytes sin firmar del dispositivo Bluetooth conectado.

number ReceiveUnsigned4ByteNumber()
: Reciba un número de 4 bytes sin firmar del dispositivo Bluetooth conectado.

list ReceiveUnsignedBytes(number numberOfBytes)
: Reciba múltiples valores de bytes sin signo desde el dispositivo Bluetooth conectado. Si numberOfBytes es menor que 0, lea hasta que se reciba un valor de byte delimitador.

Send1ByteNumber(text number)
: Envíe un número de 1 byte al dispositivo Bluetooth conectado.

Send2ByteNumber(text number)
: Envíe un número de 2 bytes al dispositivo Bluetooth conectado.

Send4ByteNumber(text number)
: Envíe un número de 4 bytes al dispositivo Bluetooth conectado.

SendBytes(list list)
: Envíe una lista de valores de bytes al dispositivo Bluetooth conectado.

`SendText(text text)`
Enviar texto al dispositivo Bluetooth conectado.

BluetoothServer

Componente del servidor Bluetooth

Propiedades

`Available: boolean`
Indica si Bluetooth está disponible en el dispositivo Android.

`CharacterEncoding: text`
La codificación de caracteres para usar al enviar y recibir texto.

`DelimiterByte: number`
El byte delimitador que se utiliza al pasar un número negativo para el parámetro numberOfBytes al llamar a ReceiveText, ReceiveSignedBytes o ReceiveUnsignedBytes.

`Enabled: boolean`
Indica si Bluetooth está habilitado.

`HighByteFirst: boolean`
Si los números de 2 y 4 bytes se deben enviar y recibir primero con el byte alto (o el más significativo). Verifique la documentación del dispositivo con el que su aplicación se comunicará para la configuración adecuada. Esto también se conoce como big-endian.

`IsAccepting: boolean`
Indica si este componente BluetoothServer acepta una conexión entrante.

`IsConnected: boolean`
Indica si se ha realizado una conexión Bluetooth.

Eventos

`ConnectionAccepted()`
Indica que se ha aceptado una conexión bluetooth.

Métodos

`AcceptConnection(text serviceName)`
Acepte una conexión entrante con el perfil de puerto serie (SPP).

`AcceptConnectionWithUUID(text serviceName, text uuid)`
Acepte una conexión entrante con un UUID específico.

`number BytesAvailableToReceive()`
Devuelve una estimación de la cantidad de bytes que se pueden recibir sin bloquear

`Disconnect()`
Desconectarse del dispositivo Bluetooth conectado.

`number ReceiveSigned1ByteNumber()`
Reciba un número de 1 byte firmado desde el dispositivo Bluetooth conectado.

`number ReceiveSigned2ByteNumber()`
Reciba un número de 2 bytes firmado desde el dispositivo Bluetooth conectado.

`number ReceiveSigned4ByteNumber()`
Recibe un número de 4 bytes firmado del dispositivo Bluetooth conectado.

`list ReceiveSignedBytes(number numberOfBytes)`
Reciba múltiples valores de bytes firmados desde el dispositivo Bluetooth conectado. Si numberOfBytes es menor que 0, lea hasta que se reciba un valor de byte delimitador.

`text ReceiveText(number numberOfBytes)`
Recibe texto del dispositivo Bluetooth conectado. Si numberOfBytes es menor que 0, lea hasta que se reciba un valor de byte delimitador.

`number ReceiveUnsigned1ByteNumber()`
Reciba un número de 1 byte sin firmar del dispositivo Bluetooth conectado.

`number ReceiveUnsigned2ByteNumber()`
Reciba un número de 2 bytes sin firmar del dispositivo Bluetooth conectado.

`number ReceiveUnsigned4ByteNumber()`
Reciba un número de 4 bytes sin firmar del dispositivo Bluetooth conectado.

`list ReceiveUnsignedBytes(number numberOfBytes)`
Reciba múltiples valores de bytes sin signo desde el dispositivo Bluetooth conectado. Si numberOfBytes es menor que 0, lea hasta que se reciba un valor de byte delimitador.

`Send1ByteNumber(text number)`
Envíe un número de 1 byte al dispositivo Bluetooth conectado.

`Send2ByteNumber(text number)`
Envíe un número de 2 bytes al dispositivo Bluetooth conectado.

`Send4ByteNumber(text number)`
Envíe un número de 4 bytes al dispositivo Bluetooth conectado.

`SendBytes(list list)`
Envíe una lista de valores de bytes al dispositivo Bluetooth conectado.

`SendText(text text)`
Enviar texto al dispositivo Bluetooth conectado.

`StopAccepting()`
Deja de aceptar una conexión entrante.

Web

Componente no visible que proporciona funciones para las solicitudes HTTP GET, POST, PUT y DELETE.

Propiedades

`AllowCookies`
Si las cookies de una respuesta se deben guardar y usar en solicitudes posteriores. Las cookies solo son compatibles con la versión de Android 2.3 o superior.

`RequestHeaders`
Los encabezados de solicitud, como una lista de sublistas de dos elementos. El primer elemento de cada sublista representa el nombre del campo del encabezado de solicitud. El segundo elemento de cada sublista representa los valores de campo del encabezado de solicitud, ya sea un valor único o una lista que contiene múltiples valores.

`ResponseFileName`
El nombre del archivo donde se debe guardar la respuesta. Si SaveResponse es verdadero y ResponseFileName está vacío, se generará un nuevo nombre de archivo.

`SaveResponse`
Si la respuesta debe guardarse en un archivo.

`Url`
La URL para la solicitud web.

Eventos

`GotFile(text url, number responseCode, text responseType, text fileName)`
Evento que indica que una solicitud ha finalizado.

`GotText(text url, number responseCode, text responseType, text responseContent)`
Evento que indica que una solicitud ha finalizado.

Métodos

`text BuildRequestData(list list)`
Convierte una lista de sublistas de dos elementos, que representan pares de nombre y valor, en una cadena formateada como application / x-www-form-urlencoded type, adecuada para pasar a PostText.

`ClearCookies()`
Borra todas las cookies para este componente web.

`Delete()`
Realiza una solicitud HTTP DELETE utilizando la propiedad Url y recupera la respuesta.
Si la propiedad SaveResponse es verdadera, la respuesta se guardará en un archivo y se desencadenará el evento GotFile. La propiedad ResponseFileName se puede usar para especificar el nombre del archivo.
Si la propiedad SaveResponse es falsa, se desencadenará el evento GotText.

`Get()`
Realiza una solicitud HTTP GET utilizando la propiedad Url y recupera la respuesta.
Si la propiedad SaveResponse es verdadera, la respuesta se guardará en un archivo y se desencadenará el evento GotFile. La propiedad ResponseFileName se puede usar para especificar el nombre del archivo.
Si la propiedad SaveResponse es falsa, se desencadenará el evento GotText.

`text HtmlTextDecode(text htmlText)`
Decodifica el valor de texto HTML dado. Entidades de caracteres HTML como & amp ;, & lt ;, & gt ;, & apos ;, & & quot; se cambian a &, <,>, ', y ". Las entidades como & # xhhhh y & # nnnn se cambian a los caracteres apropiados.

`any JsonTextDecode(text jsonText)`
Decodifica el valor codificado JSON dado para producir un valor AppInventor correspondiente. Una lista JSON [x, y, z] decodifica a una lista (xyz), un objeto JSON con

nombre A y un valor B, (denotado como A: B encerrado entre llaves) decodifica a una lista ((AB)), que es, una lista que contiene la lista de dos elementos (AB).

`PostFile(text path)`
Realiza una solicitud HTTP POST utilizando la propiedad Url y los datos del archivo especificado.
Si la propiedad SaveResponse es verdadera, la respuesta se guardará en un archivo y se desencadenará el evento GotFile. La propiedad ResponseFileName se puede usar para especificar el nombre del archivo.
Si la propiedad SaveResponse es falsa, se desencadenará el evento GotText.

`PostText(text text)`
Realiza una solicitud HTTP POST utilizando la propiedad Url y el texto especificado.
Los caracteres del texto están codificados usando la codificación UTF-8.
Si la propiedad SaveResponse es verdadera, la respuesta se guardará en un archivo y se desencadenará el evento GotFile. La propiedad responseFileName se puede usar para especificar el nombre del archivo.
Si la propiedad SaveResponse es falsa, se desencadenará el evento GotText.

`PostTextWithEncoding(text text, text encoding)`
Realiza una solicitud HTTP POST utilizando la propiedad Url y el texto especificado.
Los caracteres del texto están codificados usando la codificación dada.
Si la propiedad SaveResponse es verdadera, la respuesta se guardará en un archivo y se desencadenará el evento GotFile. La propiedad ResponseFileName se puede usar para especificar el nombre del archivo.
Si la propiedad SaveResponse es falsa, se desencadenará el evento GotText.

`PutFile(text path)`
Realiza una solicitud HTTP PUT utilizando la propiedad Url y los datos del archivo especificado.
Si la propiedad SaveResponse es verdadera, la respuesta se guardará en un archivo y se desencadenará el evento GotFile. La propiedad ResponseFileName se puede usar para especificar el nombre del archivo.
Si la propiedad SaveResponse es falsa, se desencadenará el evento GotText.

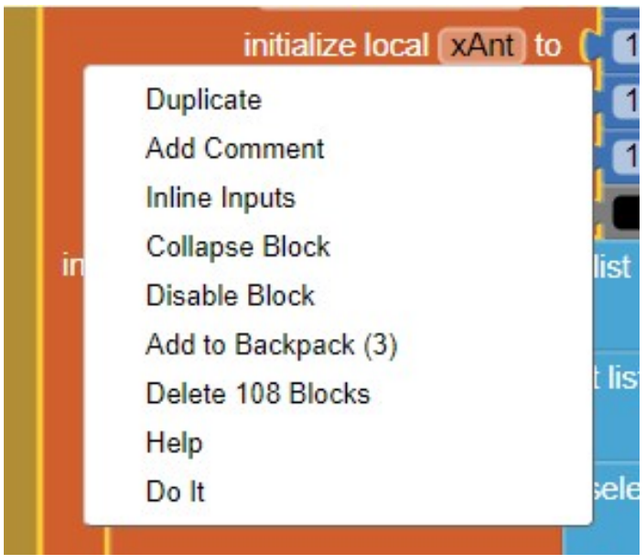
`PutText(text text)`
Realiza una solicitud HTTP PUT utilizando la propiedad Url y el texto especificado.
Los caracteres del texto están codificados usando la codificación UTF-8.
Si la propiedad SaveResponse es verdadera, la respuesta se guardará en un archivo y se desencadenará el evento GotFile. La propiedad responseFileName se puede usar para especificar el nombre del archivo.
Si la propiedad SaveResponse es falsa, se desencadenará el evento GotText.

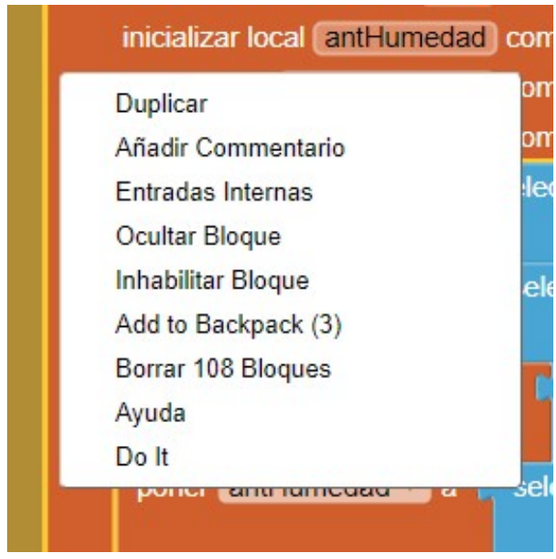
`PutTextWithEncoding(text text, text encoding)`
Realiza una solicitud HTTP PUT utilizando la propiedad Url y el texto especificado.
Los caracteres del texto están codificados usando la codificación dada.
Si la propiedad SaveResponse es verdadera, la respuesta se guardará en un archivo y se desencadenará el evento GotFile. La propiedad ResponseFileName se puede usar para especificar el nombre del archivo.
Si la propiedad SaveResponse es falsa, se desencadenará el evento GotText.

`text UriEncode(text text)`
Codifica el valor de texto dado para que pueda ser utilizado en una URL.

`any XMLTextDecode(text XmlText)`
Decodifica la cadena XML dada para producir una estructura de lista. Consulte la documentación de App Inventor en "Otros temas, notas y detalles" para obtener información.

SECTOR BLOQUES





Undo	Deshacer
Redo	Rehacer
Clean up Blocks	Limpiar los bloques
Download Blocks as Image	Exportar como Imagen
Collapse Blocks	Ocultar Bloques
Expand Blocks	Mostrar Bloques
Delete 349 Blocks	Borrar 349 Bloques
Arrange Blocks Horizontally	Ordenar Bloques Horizontalmente
Arrange Blocks Vertically	Ordenar Bloques Verticalmente
Sort Blocks by Category	Ordenar Bloques por Categoría
Paste All Blocks from Backpack (3)	Paste All Blocks from Backpack (3)
Copy All Blocks to Backpack	Copy All Blocks to Backpack
Empty the Backpack	Empty the Backpack
Enable Workspace Grid	Enable Workspace Grid
Help	Ayuda

FIN