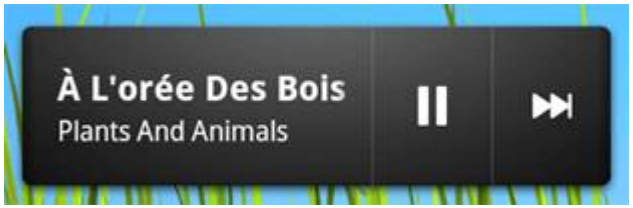
Widgets en Android

Widgets

Los **Widgets** son aplicaciones en miniaturas que pueden ser “embebidas” en otras aplicaciones (Como la pantalla HOME) y pueden recibir actualizaciones periódicas.

La pantalla de inicio (HOME), en el fondo es una aplicación que puede contener varios Widgets.Pueden existir aplicaciones como los “Launchers” que pueden contener varios widgets. A estas aplicaciones se les llama App Widget host.



Widgets

Para crear un Widget se necesita:

•**AppWidgetProviderInfo :** es un objeto que describe la metadata del widget, su periodo de actualización, el layout etc… Este es definido en XML.

•**AppWidgetProvider:** La implementación del widget que permitirá programar la lógica del widget.

•**Layout:** se debe definir un layout inicial para el widget.

•Definirlo en el manifiesto…

```
<receiver android:name="ExampleAppWidgetProvider" >
    <intent-filter>
        <action android:name="android.appwidget.action.APPWIDGET_UPDATE" />
    </intent-filter>
    <meta-data android:name="android.appwidget.provider"
               android:resource="@xml/example_appwidget_info" />
</receiver>
```

Widgets

•onUpdate() : Es llamado cada x tiempo. También se llama por primera vez cuando se agrega el widget a la pantalla.

•onDeleted(Context, int[]): Se llama cada vez que el widget se destruye.

•onEnabled(Context): Es llamado cuando se agrega la primera instancia del Widget. Los widgets puede ser agregados varias veces.

•onDisabled(Context): Es llamado cuando la última instancia de un widget se destruye.

•onReceive(Context, Intent): Es llamado cada vez que hay un Broadcast, y antes de los métodos anteriores.

Widgets

•onUpdate() : Es llamado cada x tiempo. También se llama por primera vez cuando se agrega el widget a la pantalla.

•onDeleted(Context, int[]): Se llama cada vez que el widget se destruye.

•onEnabled(Context): Es llamado cuando se agrega la primera instancia del Widget. Los widgets puede ser agregados varias veces.

•onDisabled(Context): Es llamado cuando la última instancia de un widget se destruye.

•onReceive(Context, Intent): Es llamado cada vez que hay un Broadcast, y antes de los métodos anteriores.

Ejemplo

1.Crear una actividad comun

1.Crear un widget

```
public class MiWidget extends AppWidgetProvider {
@Override
public void onUpdate(Context context,
        AppWidgetManager appWidgetManager,
        int[] appWidgetIds) {
//Actualizar el widget
//...
}
}
```

Ejemplo

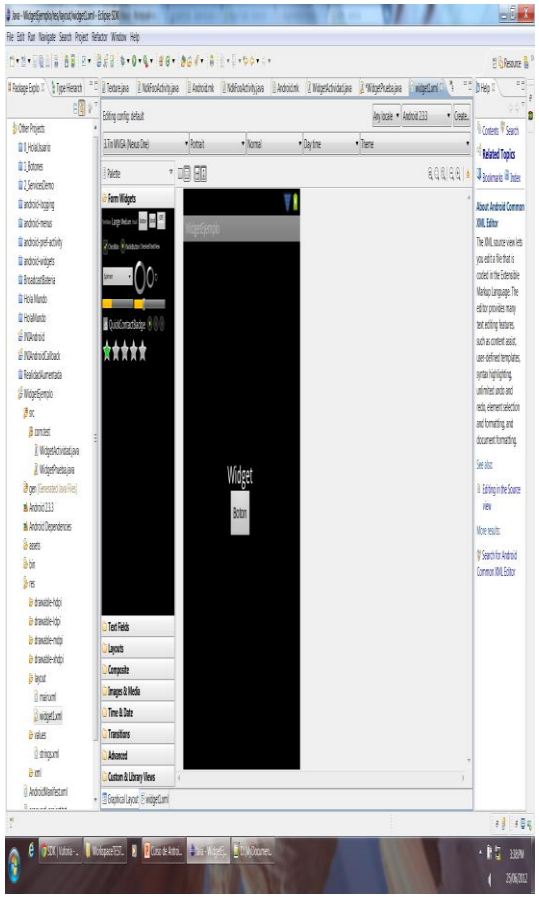
3. Definir manifiesto

```
<receiver android:name="com.test.MiWidget" android:label="Mi Primer Widget">
<intent-filter>
<action android:name="android.appwidget.action.APPWIDGET_UPDATE" />
<action android:name="com.test.ACTUALIZAR" />
</intent-filter>
<meta-data
android:name="android.appwidget.provider"
android:resource="@xml/miwidget_wprovider" />
</receiver>
```

Ejemplo 2

1.Crear una actividad y un widget
2.En el manifiesto del widget colocar un intentfilter tipo:
<action android:name="*com.test.ACTUALIZAR*" />

1.Crear un nuevo widget y crear un layout así:



Ejemplo 2

```
public void onReceive(Context context, Intent intent)
{super.onReceive(context, intent);
if (WIDGET_UPDATE.equals(intent.getAction())) {
//Obtenemos el ID del widget a actualizar
int widgetId = intent.getIntExtra(
AppWidgetManager.EXTRA_APPWIDGET_ID,
AppWidgetManager.INVALID_APPWIDGET_ID);

//Obtenemos el widget manager de nuestro contexto
AppWidgetManager widgetManager =
AppWidgetManager.getInstance(context);

//Actualizamos el widget
if (widgetId != AppWidgetManager.INVALID_APPWIDGET_ID) {
actualizarWidget(context, widgetManager, widgetId);
}
}
```

Ejemplo 2

1.En el onUpdate() del Widget colocar:

public void onUpdate(Context context, AppWidgetManager appWidgetManager, **int[] appWidgetIds)**
{

```
//Iteramos la lista de widgets en ejecución
for (int i = 0; i < appWidgetIds.length; i++)
{
//ID del widget actual
int widgetId = appWidgetIds[i];

//Actualizamos el widget actual
actualizarWidget(context, appWidgetManager, widgetId);
}
```

Ejemplo 2

```
Public static void actualizarWidget(Context context, AppWidgetManager appWidgetManager, int widgetId)
{
//Obtenemos la lista de controles del widget actual
RemoteViews controles =   new RemoteViews(context.getPackageName(), R.layout.widget1);
//Asociamos los 'eventos' al widget
Intent intent = new Intent(WIDGET_UPDATE);
intent.putExtra(
    AppWidgetManager.EXTRA_APPWIDGET_ID, widgetId);
PendingIntent pendingIntent =
PendingIntent.getBroadcast(context, widgetId,
intent, PendingIntent.FLAG_UPDATE_CURRENT);
controles.setOnClickPendingIntent(R.id.boton, pendingIntent);
//Obtenemos la hora actual
Calendar calendario = new GregorianCalendar();
String hora = calendario.getTime().toLocaleString();
//Actualizamos la hora en el control del widget
controles.setTextViewText(R.id.texto, hora);
//Notificamos al manager de la actualización del widget actual
appWidgetManager.updateAppWidget(widgetId, controles);
}
```

Ejemplo 2

Ejercicio de integración.

Crear un Widget con dos botones. El primer botón actualizará la hora del Widget, el segundo botón irá abrirá una pantalla que mostrará:

1.El tiempo en ese instante

2.El estado de la batería en ese instante.

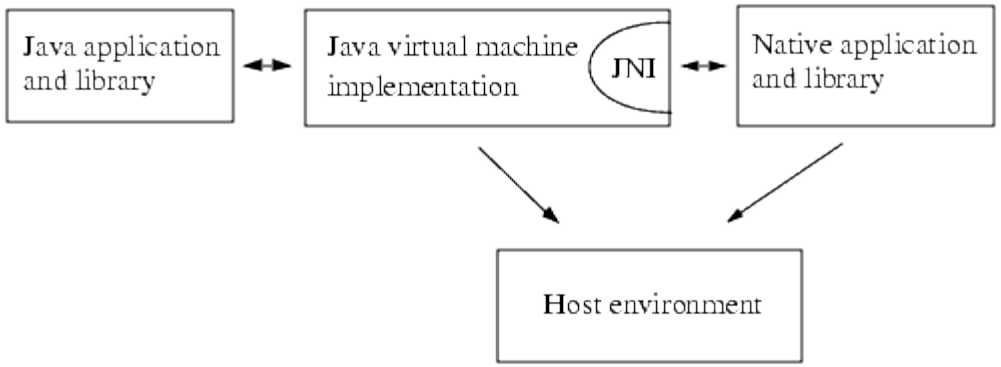
3.Dejar un boton libre para Realidad Aumentada

JNI en Android

JNI

•El JNI se usa para escribir métodos nativos que permitan solventar situaciones en las que una aplicación no puede ser enteramente escrita en Java, como por ejemplo en el caso de que la biblioteca estándar de clases no proporcione soporte para funcionalidades dependientes de la plataforma.

•También se usa para modificar programas existentes escritos en algún otro lenguaje, permitiéndoles ser accesibles desde aplicaciones Java.



JNI

Cuando No Usarlo

- **Cuando se puede interoperar dos librerías en dos lenguajes mediante TCP/IP. Ej. Por XML-RPC, Web-REST etc..**
- **Cuando el rendimiento no es prioritario.**
- **Cuando la portabilidad es necesaria.**

Cuando Usarlo:

- **Cuando se desea implementar código a bajo nivel. Por ejemplo aplicaciones 3-D.**
- **Cuando se desean soportar operaciones específicas que no son soportadas por Java.**

JNI en Android

- **Android soporta JNI, sin embargo se tiene que tener cuidado ya que vuelve la JVM suceptible al código en C.**

- **JNI está soportado por el NDK de Android**

- Descargar y Descomprimir el archivo: http://developer.android.com/tools/sdk/ndk/index.html
- **Poner la carpeta android-ndk-r* en el Path del sistema.**

JNI en Android

- **Android MK file: tiene la configuración del JNI, cómo el compilador ndk debe tratar los archivos… versión de APIS, etc jni/Android.mk**

- **Código en C o C++ . Ej. Micodigo.c**

JNIEXPORT jstring JNICALL
Java_com_test_NdkFooActivity_invokeNativeFunction(JNIEnv env, jobject javaThis) {*

JNI en Android

- **Las Librerías se cargan con:**

```
// Cargar la libreria que hace matching con: jni/Android.mk
static {
System.loadLibrary("ndkfoo");
}
```

- **Llamadas a librerías Nativas en Java:**
  - ***Private native String obtenerDatos();***
  - ***Private native void iniciarEntorno();***

JNI en Android

1.Descargar CYGWIN en la dirección:

http://www.cygwin.com/setup.exe y seleccionar "Install from the Internet!“

1.All -> Devel -> "make: The GNU version of the 'make' utility“

1.Configurar las variables de entornos:
PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/libexec:/System/Library/CoreServices:/Developer/usr/bin:
Development/Android/android-ndk-r7:export PATH

1.Probar el comando ndk-build

JNI en Android

1.Descargar CYGWIN en la dirección:

http://www.cygwin.com/setup.exe and select "Install from the Internet!“

1.All -> Devel -> "make: The GNU version of the 'make' utility“

1.Configurar las variables de entornos:
PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/libexec:/System/Library/CoreServices:/Developer/usr/bin:
Development/Android/android-ndk-r7:export PATH

1.Probar el comando ndk-build

JNI en Android

1.Crear una actividad sencilla y declarar el método:

private native String invokeNativeFunction();

1.Cargar la librería:

// Cargar la libreria que hace matching con: jni/Android.mk
static {
System.*loadLibrary("ndkfoo");*
}

1.Crear un Boton y en el onclicListener capturar el valor de la invocación ; y luego visualizarlo con un TOAST

Ej: String miinfo = invokeNativeFunction();

JNI en Android

*Crear el archivo N**dkfoo.c***

```
#include <string.h>
#include <jni.h>

JNIEXPORT jstring JNICALL
Java_com_test_NdkFooActivity_invokeNativeFunction(JNIEnv* env, jobject
javaThis) {
  return (*env)->NewStringUTF(env, "Hola del codigo en C!");
}
```

JNI en Android

Agregar en el Android.mk

```
LOCAL_PATH := $(call my-dir)

include $(CLEAR_VARS)

# Here we give our module name and source file(s)
LOCAL_MODULE    := ndkfoo
LOCAL_SRC_FILES := Ndkfoo.c

TARGET_PLATFORM := android-10

include $(BUILD_SHARED_LIBRARY)
```

JNI en Android

1.Irse a la carpeta del projecto e invocar:
ndk-build