Auto start a Delphi XE5 Android App after boot

How to use a broadcast receiver to automatically start a Delphi XE5 App after BOOT_COMPLETED on an Android device.

In good-old DOS days we could start an application after boot by adding it to autoexec.bat. Under Windows you can add a program to startup using the registry or view startup programs with msconfig. But how does this work under Android?

Automatically starting a Delphi XE5 Android app after the device has booted up can be done by creating a broadcast receiver that listens to the BOOT_COMPLETED broadcast intent.

http://developer.android.com/reference/android/content/BroadcastReceiver.html

This article descibes the steps necessary to create such a broadcast receiver with Delphi XE5:

- create a new Delphi XE5 Android project
- set uses permissions to receive boot completed
- modify AndroidManifest.template.xml to let the Android system know you have a broadcast receiver
- write some Java code
- add it to the classes.dex
- use this new classes.dex in your project
- run the app on device

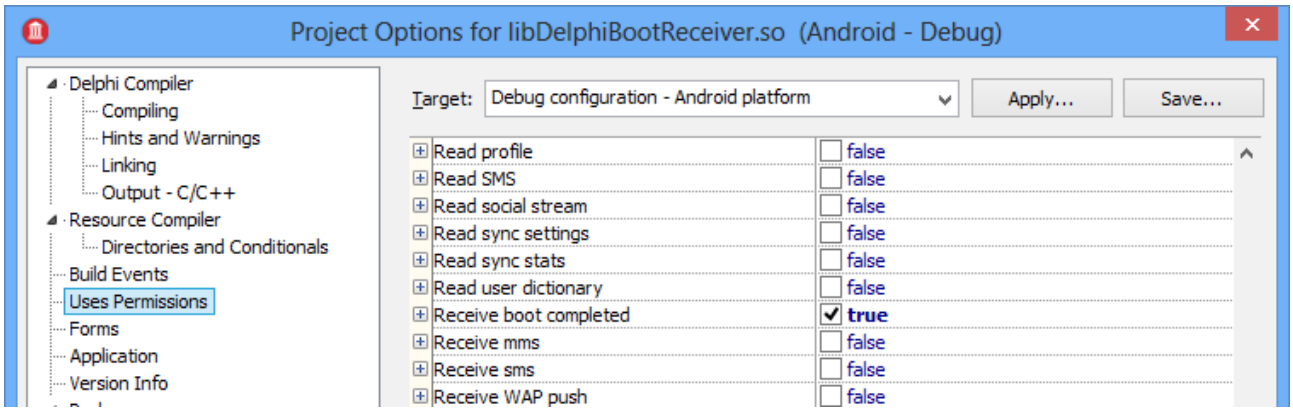
Create a new Delphi XE5 Android project

This step should be easy enough, File | New | Firemonkey Mobile Application. Just save it someplace that you can also reach with the command prompt console.

Set uses permissions

All Android apps must register which permissions they need in the Android Manifest. The manifest is read by the Android system when installing the app package and the user is notified of the requested permissions. If you forgot to list a permission you need, access to that function will be denied by the system. In this case we need the receive boot completed permission.

Open Project | Options and uses permissions to set this permission to true.



Modify AndroidManifest.template.xml

The Android Manifest template is used by the Delphi IDE to re-create the manifest for your Android package (.apk), each time you deploy the project to an apk. If you want to modify the Manifest, this is the place to do so.

In this manifest we will notify the Android system that we have a receiver for the broadcast intent BOOT_COMPLETED. The new code is in black, existing stuff in blue:

```
</activity>
        <receiver android:name="com.embarcadero.firemonkey.notifications.FMXNotificationAlarm" />
        <receiver android:name="com.dannywind.delphi.BootReceiver"
                  android:permission="android.permission.RECEIVE_BOOT_COMPLETED">
           <intent-filter>
                <action android:name="android.intent.action.BOOT_COMPLETED" />
                <category android:name="android.intent.category.DEFAULT" />
          </intent-filter>
        </receiver>
    </application>
</manifest>
<!-- END_INCLUDE(manifest) -->
```

Write some Java code

Unfortunately the current version of Delphi does not have a pre-defined receiver that we can use to autostart the app. To make this possible we have to write a bit of Java code and include this into the classes.dex file of your Delphi Android app.

The Java code is as follows. Use BootReceiver.java as the file name:

```
package com.dannywind.delphi;

import android.app.Activity;
import android.content.BroadcastReceiver;
import android.content.Intent;
import android.content.Context;
import android.util.Log;

public class BootReceiver extends BroadcastReceiver
{

        @Override
        public void onReceive(Context context, Intent intent)
        {
           Intent launchintent = new Intent();
           launchintent.setClassName(context, "com.embarcadero.firemonkey.FMXNativeActivity");
           launchintent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
           context.startActivity(launchintent);
        }

}
```

The context variable used to start the activity is actually your app's context. We're essentially starting the app, from within the app as a new task. Because the native activity in a default Delphi app has a default name, we can just start the com.embarcadero.firemonkey.FMXNativeActivity of the current app.

Add it to the classes.dex

Delphi has a pre-compiled classes.dex which is put into any Delphi Android package in the deployment step. We are going to add this Java code to the classes.dex, and use the deployment manager to replace the default classes.dex with our modified version.

In your project directory create a subdirectory java\src\com\dannywind\delphi and place the BootReceiver.java code there. Build the Java code using the following batch file code:

```
@echo off
echo.
echo Compiles your Java code into classes.dex
echo Verified to work in Delphi XE5 Update 1 and 2
echo.
echo Place this batch in a java folder below your project (project\java)
echo Place the source in project\java\src\com\dannywind\delphi
echo If your source file location or name is different, please modify it below.
echo.

setlocal

set ANDROID_JAR="C:\Users\Public\Documents\RAD Studio\12.0\PlatformSDKs\adt-bundle-windows-x86-
20130522\sdk\platforms\android-17\android.jar"
set DX_LIB="C:\Users\Public\Documents\RAD Studio\12.0\PlatformSDKs\adt-bundle-windows-x86-
20130522\sdk\build-tools\android-4.2.2\lib"
set EMBO_DEX="C:\Program Files (x86)\Embarcadero\RAD Studio\12.0\lib\android\debug\classes.dex"
set PROJ_DIR=%CD%
set VERBOSE=0
set JAVASDK="C:\Program Files\Java\jdk1.7.0_25\bin"
set DX_BAT="C:\Users\Public\Documents\RAD Studio\12.0\PlatformSDKs\adt-bundle-windows-x86-
20130522\sdk\build-tools\android-4.2.2\dx.bat"

echo.
echo Compiling the Java source files
echo.
pause
mkdir output 2> nul
mkdir output\classes 2> nul
if x%VERBOSE% == x1 SET VERBOSE_FLAG=-verbose
%JAVASDK%\javac %VERBOSE_FLAG% -Xlint:all -classpath %ANDROID_JAR% -d output\classes -source 1.6
-target 1.6 src\com\dannywind\delphi\BootReceiver.java

echo.
echo Creating jar containing the new classes
echo.
pause
mkdir output\jar 2> nul
if x%VERBOSE% == x1 SET VERBOSE_FLAG=v
%JAVASDK%\jar c%VERBOSE_FLAG%f output\jar\test_classes.jar -C output\classes com


echo.
echo Converting from jar to dex...
echo.
pause
mkdir output\dex 2> nul
if x%VERBOSE% == x1 SET VERBOSE_FLAG=--verbose
call %DX_BAT% --dex %VERBOSE_FLAG% --output=%PROJ_DIR%\output\dex\test_classes.dex
--positions=lines %PROJ_DIR%\output\jar\test_classes.jar

echo.
echo Merging dex files
echo.
pause
%JAVASDK%\java -cp %DX_LIB%\dx.jar com.android.dx.merge.DexMerger %PROJ_DIR%\output\dex\classes.dex
%PROJ_DIR%\output\dex\test_classes.dex %EMBO_DEX%

echo.
echo Now use output\dex\classes.dex instead of default classes.dex
echo And add broadcastreceiver to AndroidManifest.template.xml
echo.

:Exit

endlocal
```

The above batch file was originally written by Brian Long for merging your own Java scripts with

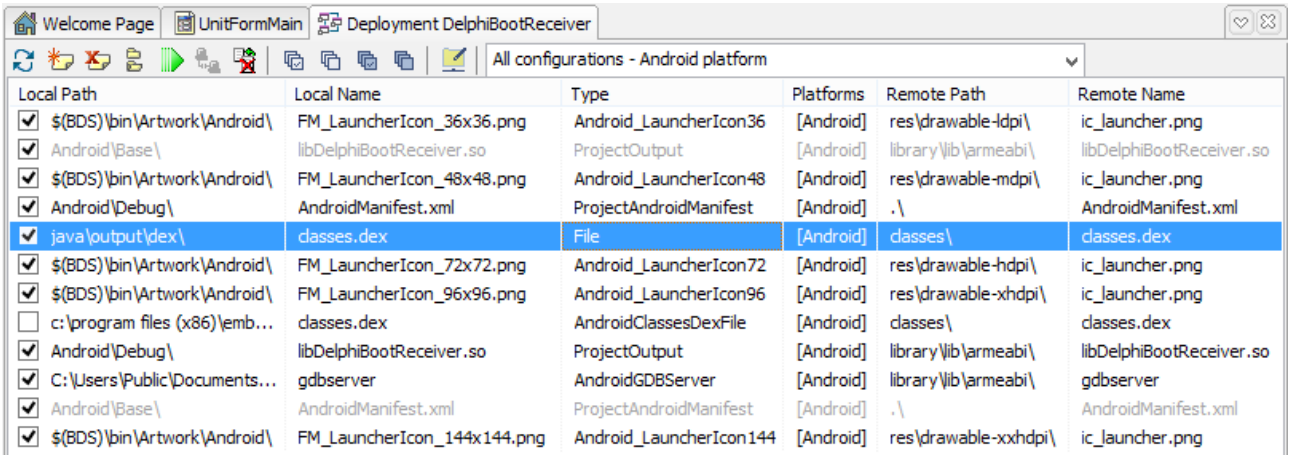
classes.dex. Thanks to Brian for making it available. I have made some minor modifications to it.

Place the build.bat file into the java subdirectory under your project directory project\java, open a command prompt and just run build.bat.

When running the batch file you'll see a warning “bootstrap class path not set”. We are using the Java SDK 1.7 that is bundled with Delphi XE5 to build 1.6 compatible source code for classes.dex. If you do have 1.6 installed you can add this option to get rid of the warning “-bootclasspath C:\jdk1.6.0\lib\rt.jar”, but you can also just ignore it safely as it will not in any way interfere with our snippet of Java code.

Use this new classes.dex

Using the deployment manager we disable the default classes.dex, and add our new expanded classes.dex file with the add files button.



Run the app on device

Now run the app on the Android device. If it runs succesfully, just let it run, and switch the device off with a long press on the on/off button. Android will ask you if you want to switch it off, which of course you do. Now turn it back on and your Delphi app will autorun after the boot sequence is completed.

Tips

The Delphi XE5 IDE really wants you to use the default classes.dex file, so it will re-enable it every time you switch deployment configuration. If you get a “App has stopped” message, please check first if Delphi hasn't re-enabled the default classes.dex file. You can also confirm this by scanning logcat for “Didn't find class “com.dannywind.delphi.BootReceiver””.

When you use Project | Deploy, Delphi creates an .apk file on your local harddisk. It does not actually deploy to your Android device. To deploy to device, just run it.

Your autorun Delphi app will not receive a boot completed event if it is in a stopped state. This is a feature that was added to Android 3.1 to improve security (http://developer.android.com/about/versions/android-3.1.html#api). What this means is the user must run the app at least once, as this will get it out of lockdown. Also if you force stop the app using manage applications it will again be in a stopped state and will no longer autorun. Just run the app once again and autorun will work again. Please note that closing the app with the recent apps button and then swiping it up; does not equal a force close and does not put it in a stopped state. If you close your app this way it will succesfully autorun after reboot.

To make autorun more robust you could write a separate service in Java to start the app using its full name. The broadcast receiver code for this service would be similar to the above code, except for this line of code:

```
launchintent.setClassName("com.embarcadero.DelphiBootReceiver",
"com.embarcadero.firemonkey.FMXNativeActivity");
```

Of course there are also apps available in the play store that do just that.