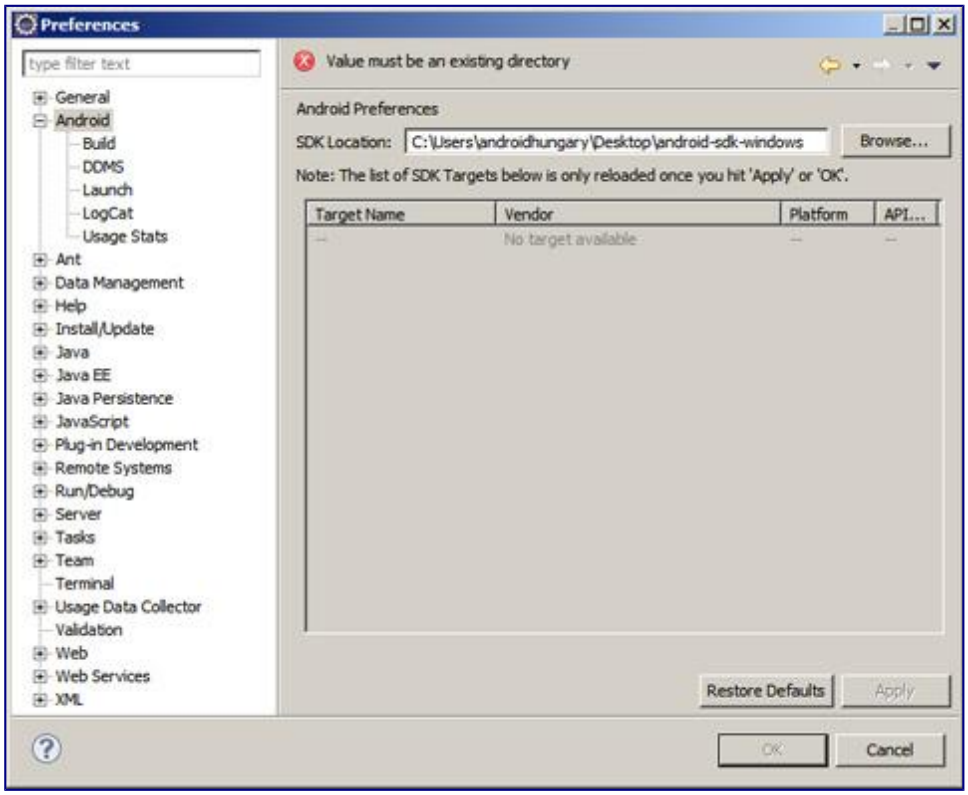
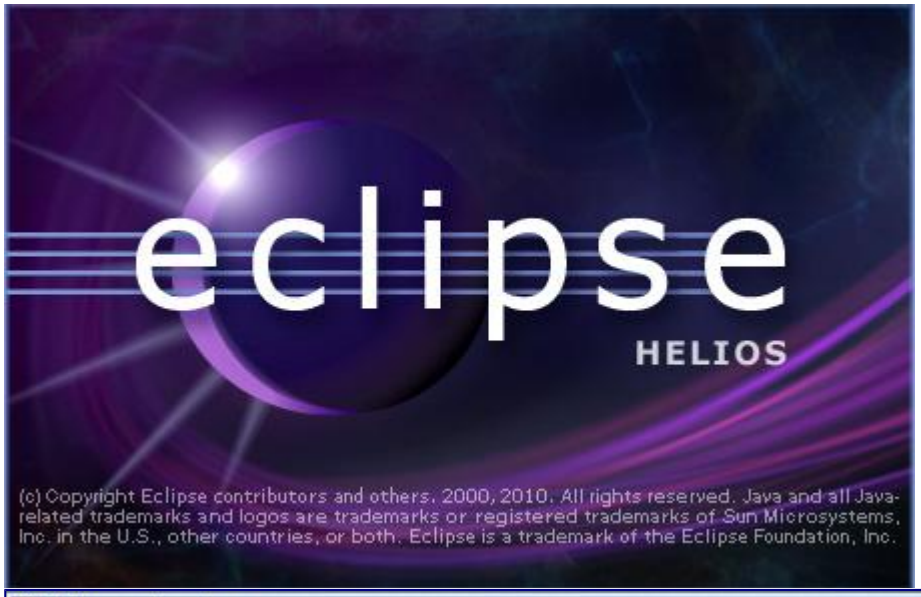
Android fejlesztői lecke – A fejlesztői környezet áttekintése
http://androidhungary.com/?s=Android+honos%C3%ADt%C3%A1s&x=23&y=10
2010. szeptember 14. | 13 hozzászólás | Kategória: Fejlesztés

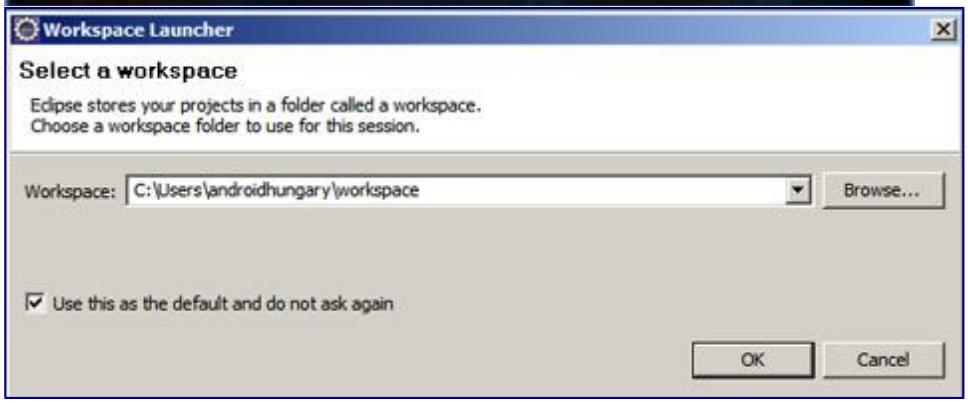
Az informatika területén belül a mobiltelefónia az egyik legjobban fejlődő ágazat, melyek közül a két legdinamikusabb húzóerő az iOS és az Android OS.
Egy kezdő fejlesztő számára mindig kérdéses, hogy mit is kezdjen el először tanulni.
Az imént említett mobiltelefonos platformok közül talán az Androidra lehet a leginkább platformfüggetlenül fejleszteni – köszönhetően a Java nyelvnek.
A korábban megjelent Android honosítással kapcsolatos bemutatókhoz hasonlóan, most is egy több részes cikksorozatot szeretnénk elindítani azok számára, akik kicsit szeretnének jobban elmélyülni az Android fejlesztés, programozás világában.
A fejlesztői környezet (avagy IDE – Integrated Development Environment, esetünkben Eclipse), maga is Java nyelven íródott, amely elérhető a legnépszerűbb operációs rendszerekre, Windowsra, Mac OS X-re, és különböző Linux disztribúciókra egyaránt. Amennyiben Android fejlesztővé szeretnénk válni, mindenképpen meg kell ismerkednünk az Android fejlesztőknek szánt oldalával, hiszen az itt található dokumentációk óriási segítségünkre lesznek a tanulás, problémamegoldás folyamán.
Fontos megemlíteni, hogy az iOS platformmal ellentétben, nem szükséges regisztrált fejlesztőnek lennünk ahhoz, hogy az alkalmazásunkat a telefonunkon is kipróbálhassuk, vagy az interneten .apk formátumban terjesszük azt.
A cikksorozat első részében a fejlesztői környezet telepítését és egy példaprogram elkészítését mutatjuk be Windows-os környezetben.
Ehhez a következő lépéseket kell végrehajtanunk.

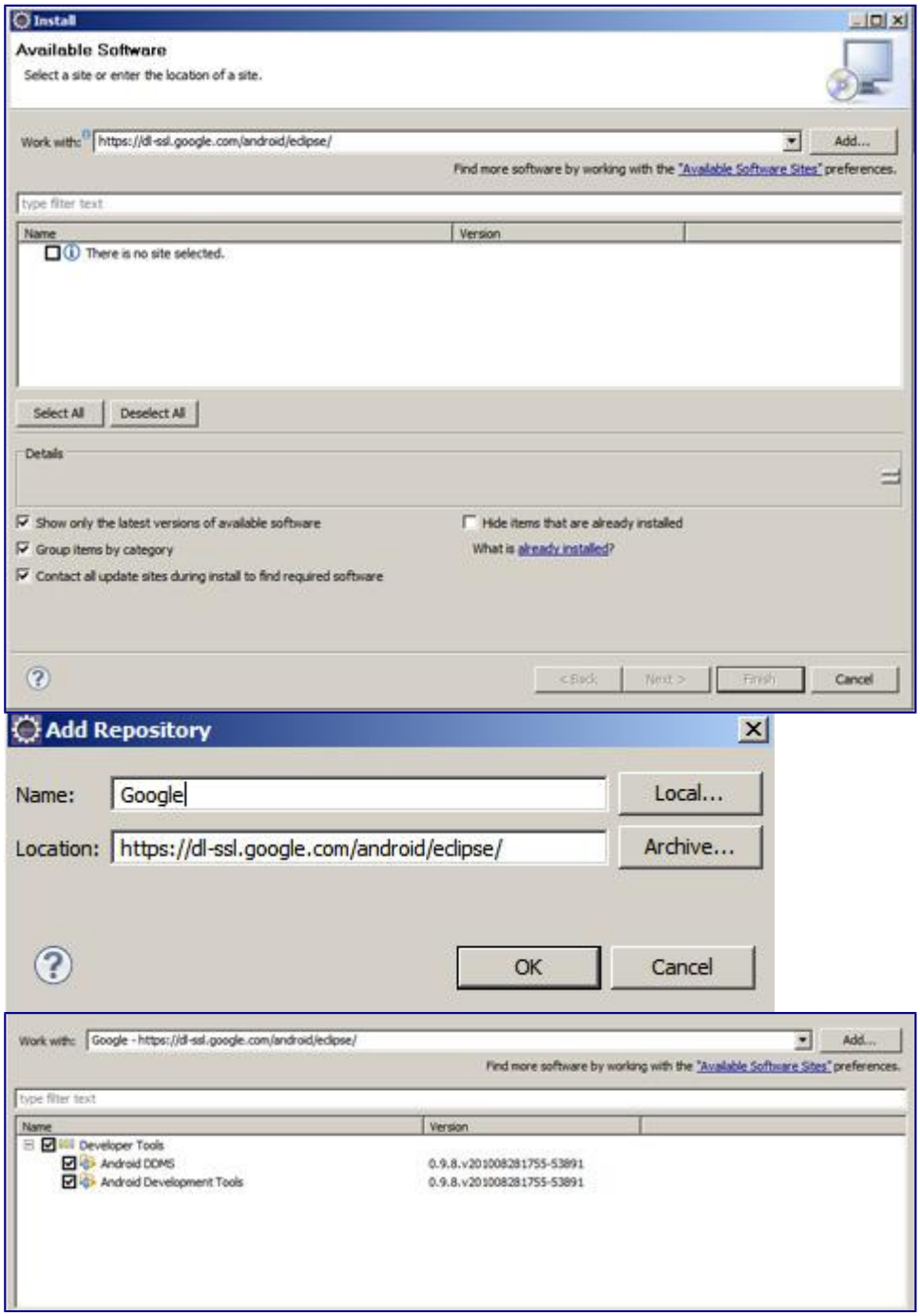
- Amennyiben gépünkön még nincs fent a Java Runtime Environment, úgy letöltés után telepítsük innen.
- Ezután töltsük le az Eclipse Java EE alkalmazást.
- Miután letöltődött, bontsuk ki a .zip fájl tartalmát, pl. az asztalra.
- Majd installáljuk az Android SDK-t, és ugyancsak bontsuk ki, az egyszerűség kedvéért ezt is az asztalra.



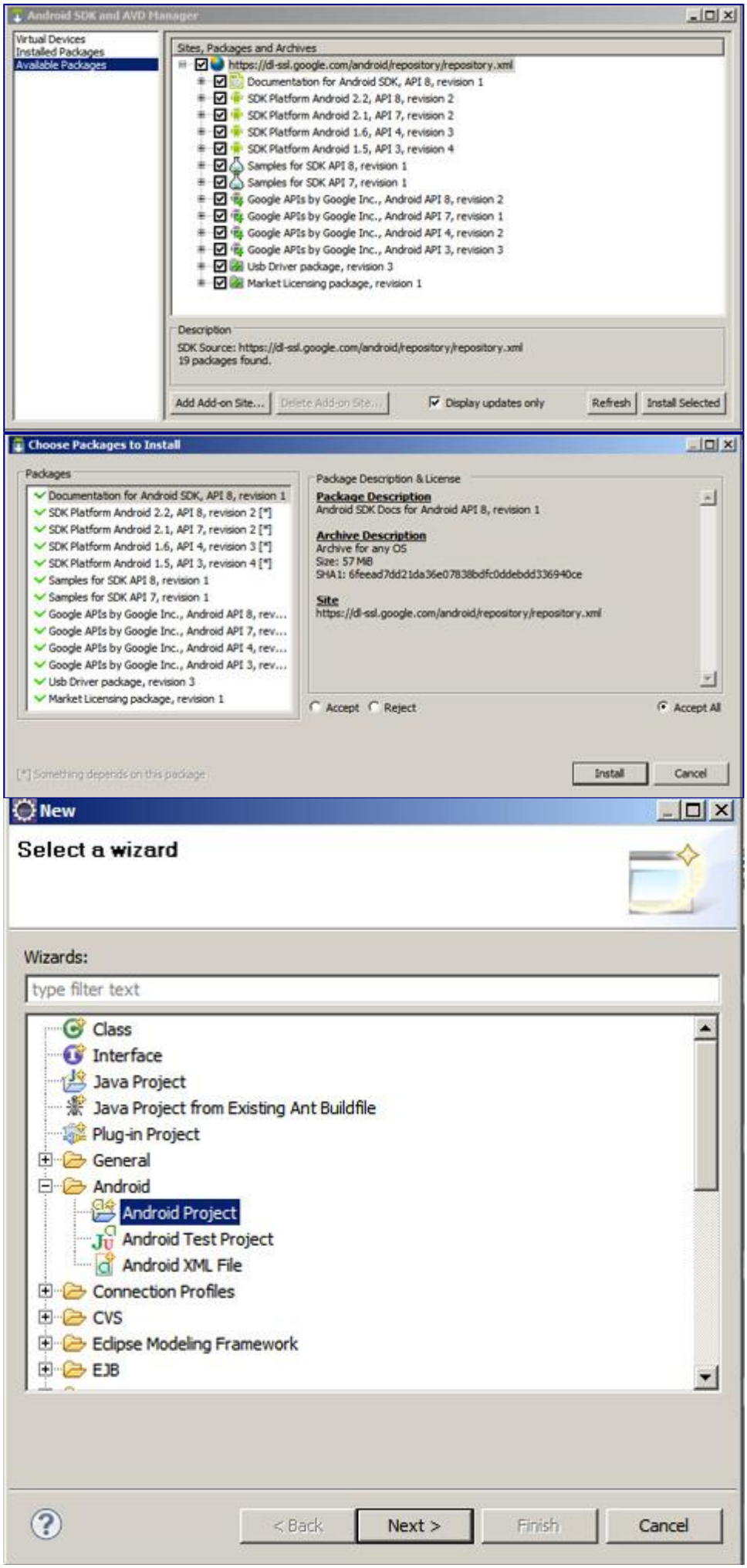
Indítsuk el az imént kicsomagolt Eclipse programot és kattintsunk a Help/Install new software menüre.
A "work with" opcióhoz írjuk be a https://dl-ssl.google.com/ hivatkozást, majd nyomjunk az Add gombra.
A folyamatot az OK gomb lenyomásával folytassuk, aminek hatására meg is jelent a Developer Tools, ami mellett jelöljük be a checkbox-ot, majd kattintsunk a Next gombra.
A következő ablakban fogadjuk el a felhasználási feltételeket, majd pedig a Finish gomb lenyomásával véglegesítsük a folyamatot.





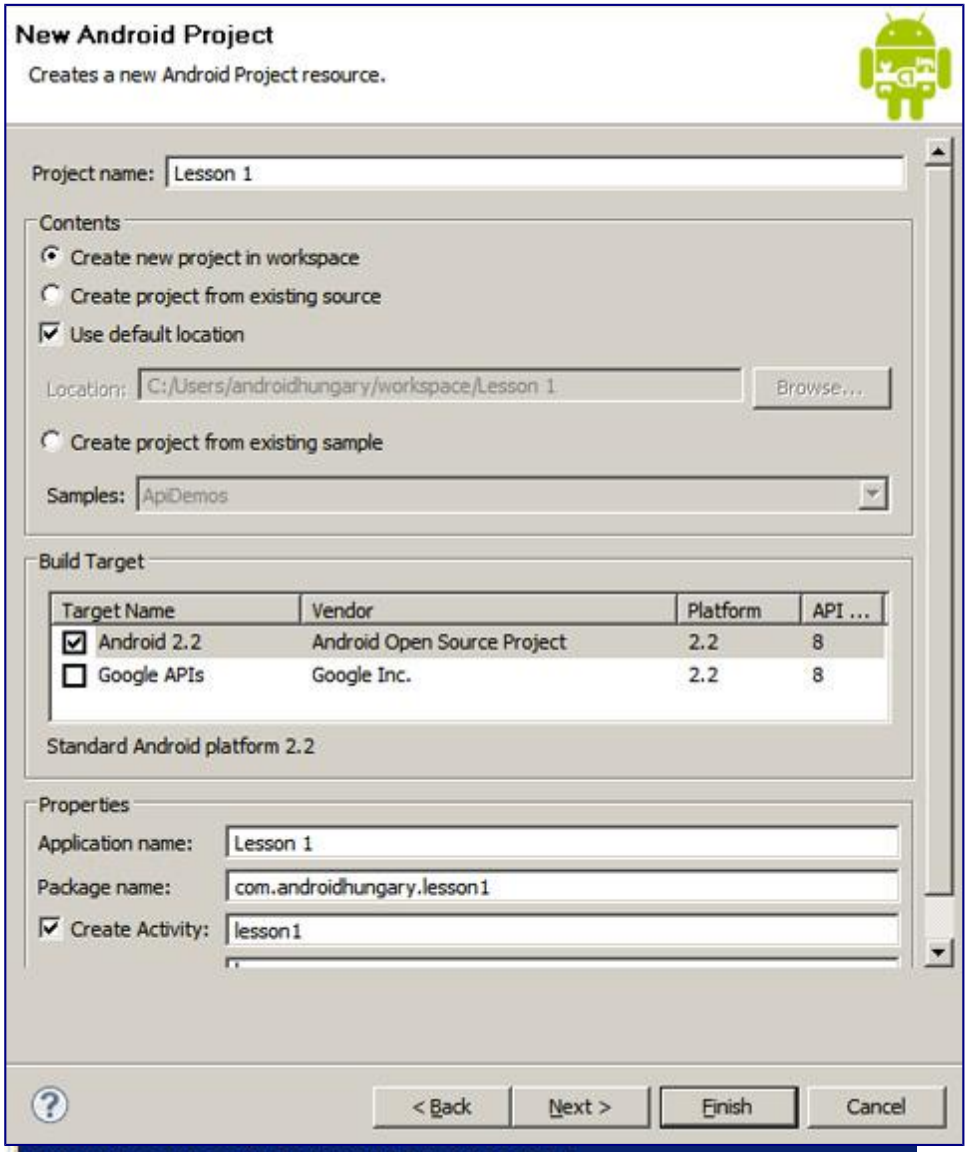


Miután a szükséges mezőket kitöltöttük, nyomjunk a Finish gombra. A baloldali struktúrában nyissuk meg a Lesson 1 – src – com.androidhungary.lesson1 – lesson1.java fájlt.
Írjuk be a kódot, és kattintsunk a Build menüre és válasszuk ki, hogy Android Application.
Amennyiben rögtön a telefonunkra szeretnénk telepíteni az alkalmazást, csatlakoztassuk a készülékünket USB kábelen, (ne felejtsük el bekapcsolni a Settings/Applications/Unknown sources, és Settings/Applications/
Ha viszont virtuális eszközön szeretnénk tesztelni, adjuk hozzá az SDK-ban, majd utána indítsuk el.
Az előbbi rövid bemutató alkalmazással el is készítettük első Android OS alatt futó programunkat.



Miután a szükséges mezőket kitöltöttük, nyomjunk a Finish gombra. A baloldali struktúrában nyissuk meg a Lesson 1 – src – com.androidhungary.lesson1 – lesson1.java fájlt.

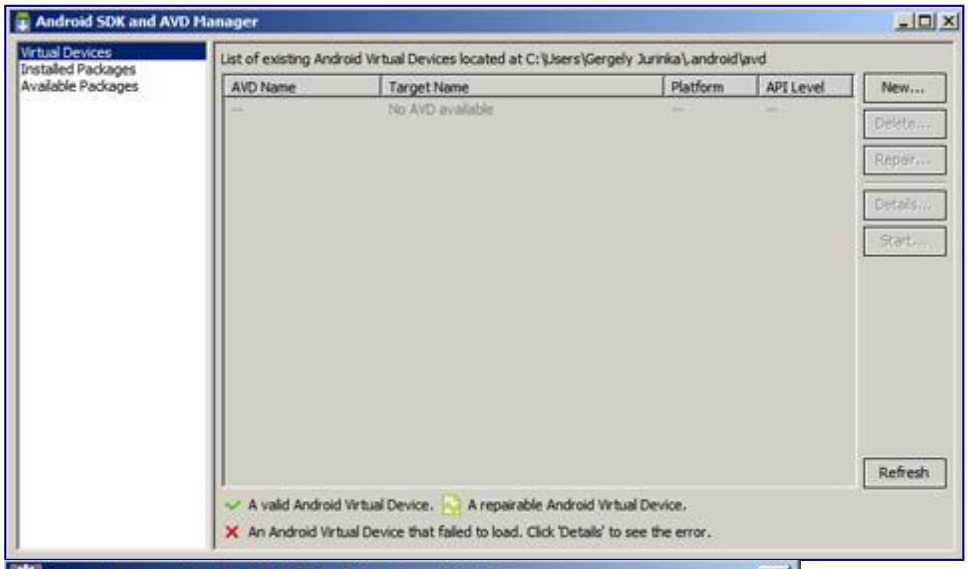
Írjuk be a kódot, és kattintsunk a Build menüre és válasszuk ki, hogy Android Application.
Amennyiben rögtön a telefonunkra szeretnénk telepíteni az alkalmazást, csatlakoztassuk a készülékünket USB kábelen, (ne felejtsük el bekapcsolni a Settings/Applications/Unknown sources, és Settings/Applications/Development/USB debugging beállításokat).
Ha viszont virtuális eszközön szeretnénk tesztelni, adjuk hozzá az SDK-ban, majd utána indítsuk el.
Az előbbi rövid bemutató alkalmazással el is készítettük első Android OS alatt futó programunkat.

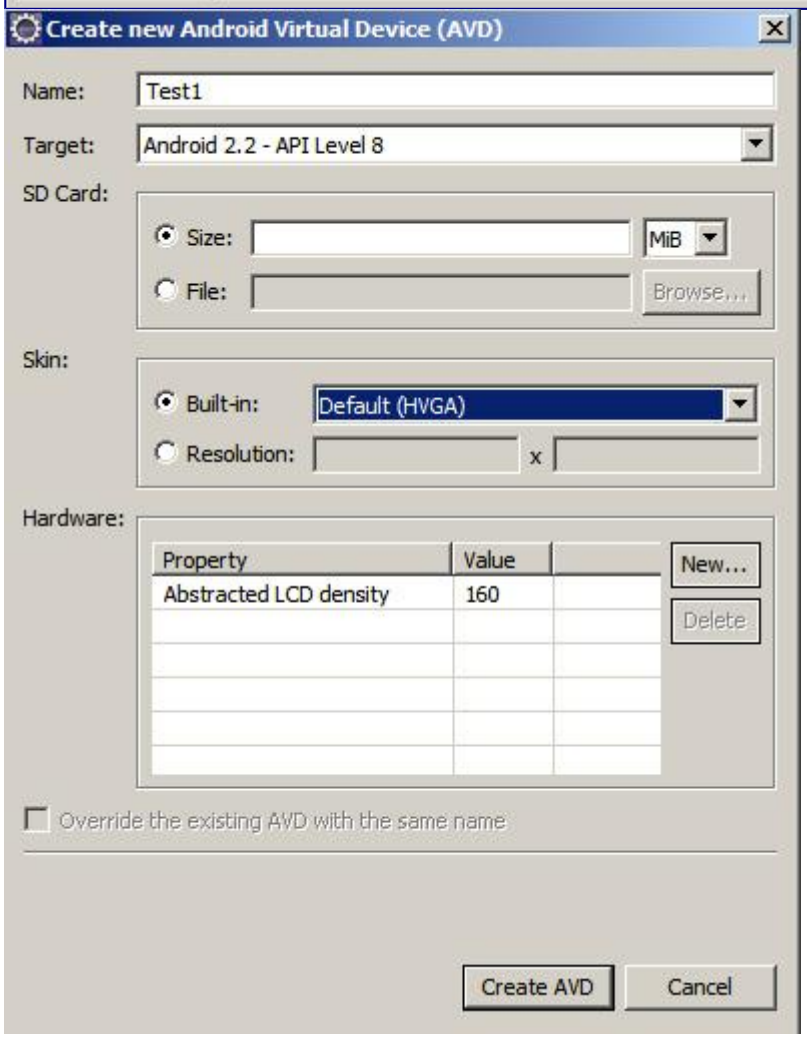



```
package com.androidhungary.lesson1;

import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;

public class lesson1 extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        TextView tv = new TextView(this);
        tv.setText("Hello, Android");
        setContentView(tv);
    }
}
```





Android honosítás 1. rész – Az android csomagjai
2010. május 14. | 42 hozzászólás | Kategória: Android OS, Bemutató, Fejlesztés

Az Android operációs rendszerek honosításának menetéről most egy háromrészes cikksorozat készül, melynek első része az android apk csomagjainak felépítését hivatott bemutatni.
Lehet, hogy sokaknak kicsit száraznak fog tűnni, de azért vágjunk is bele!
Sokszor hallottam már, hogy semmi extra nincs az Androidban, hiszen ugyanúgy Java csomagokból áll, mint a sima Javás telefonokra telepíthető programok.
Ez részben igaz, az Android Javára épül, viszont nem csak egyszerű Java platformról van szó, hiszen ott van mellette maga az Android platform is.
Az operációs rendszer felépítése a következő:

- Linux kernel. Ez a nagyszerű rendszermag dobog az összes Android telefon szívében. A feladata, hogy összekösse a szoftvert a hardverrel, és kezelje az egyes eszközöket, pl.: kamera, g-szenzor, gps, stb.

- Busybox. Ez a szoftver található meg általában az embedded Linux eszközökön, pl.: routerekben, fényképezőgépekben, és mint azt példánk mutatja: telefonokban. A busybox egy kis hely- és erőforrásigényű shell alapú programgyűjtemény. Ez indítja el a telefon egyes funkcióit, mint pl. a Dalvik VM
- Dalvik. Ez a szoftver már az Android csomag része, neki köszönhető, hogy platformfüggetlenül telepíthetjük programjainkat bármilyen Android alapú készülékre. Nem más, mint egy virtuális gép, ami magát az androidot futtatja.
- Framework. Az Android felhasználói felülete. A későbbiekben sok szó lesz róla.

Innen kezd érdekesebbé válni a dolog, mert eljutottunk arra a részre ahol az egyes alkalmazások futnak.
Az Android alkalmazásai apk kiterjesztésű zip csomagokban vannak tárolva. Az apk csomagok három helyen kerülnek tárolásra.
/system/framework/ – A framework egy részét is apk csomagok képzik, nevezetesen a framework-res.apk csomag. Ez nagyon fontos a rendszer számára, mivel alapvető adatokat tartalmaz a telefon felületének elrendezéséről, a felület grafikai elemeiről, az Android rendszer szöveges állományairól, az egyes ablakok/képernyők színezetéről, és még rengeteg dologról.
/system/app/ – Itt találhatóak a telefonra gyárilag feltelepített alkalmazások, mint például a galéria, a telefon, a névjegyzék, stb.
/data/app/ – Itt pedig a Marketből, vagy egyéb forrásból telepített alkalmazások kapnak helyet.
Az apk csomagok jelentik tehát mindazt, amit a kijelzőn láthatunk.
A készülék minden képi, és szöveges eleme ezekben a csomagokban van meghatározva, tehát ha módosítani akarunk valamit, elég az apk-kat módosítani.
Ennek két jól járható útja is van, ahol az első az, ha rendelkezésre állnak az apk-k forrásai.
Ebben az esetben könnyű dolgunk van, egyszerűen elhelyezhetjük a lokalizációnkat a csomagban, és azt lefordítva remélhetőleg egy már működő apk csomagot kapunk, amit használatba is vehetünk.
Az esetek többségében viszont olyan apk-k fognak a kezeink közé kerülni, amik már módosításokat tartalmazzák, vagy nem rendelkezünk a forrásával. Ez esetben jön képbe a másik út, ki kell bontanunk, és vissza kel fejtenünk a csomagot.
Ennek mibenlétét akkor láthatjuk át, ha jobban megismerjük a csomagokat.
Mint azt már írtam, egyszerű zip állományokról van szó, tehát valamilyen archívum-kezelővel meg is tudjuk nyitni a csomagokat.
Az apk-knak három nagyon fontos része van: az AndroidManifest.xml fájl, az src mappa és a res mappa.
AndroidManifest.xml. Ez az apk lelke. Ebben a fájlban vannak meghatározva az activity-k, amik a program felületeit jelentik.
Az apk csomagoknak rengeteg féle activityjük lehet. Lehet indítható alkalmazás, widget, esetleg valamely rendszermenübe beépülő activity, amely lehet automatikusan induló, a háttérben futó alkalmazás, stb.
Jó példa erre a gallery.apk csomag, ami egyszerre tartalmazza a galériát, a fényképezőt, a videórögzítőt, és a képkeret widgetet.
Az AndroidManifest.xml fájlban határozható meg az is, hogy milyen engedélyekkel rendelkezzen a szóban forgó apk.
Ez az a lista, amit telepítéskor láthatunk, amikhez hozzáfér az alkalmazás a telefonban.
Src mappa. Ezt a mappát nem látjuk, csak a forrásban. Ha egy archívumkezelővel megnyitunk egy apk-t, classes.dex néven találunk benne egy fájlt.
Ebbe a fájlba van belekódolva a teljes src mappa tartalma. Itt kerül tárolásra a csomag Java nyelvű utasításlistája, a programok, amit a telefon futtat, ha elindítunk egy-egy alkalmazást.
Létezik mód a mappa visszafejtésére, de a lokalizációhoz nincs rá szükségünk.
Res mappa. Számunkra ez a legfontosabb része az apk-knak, itt található többek között az apk összes szövege, méghozzá nyelvenként külön-külön.
A sorozat harmadik cikke fog foglalkozni a res mappa működésével, tehát most nem fogom az egyes állományokat részletesen bemutatni.
Fontos megemlíteni viszont, hogy az archívumkezelővel kibontva itt sem fogunk több dolgot látni, mint a fájlok neveit és a képeket. Az apk-k rengeteg xml fájlt tartalmaznak, viszont mindet titkosítva találjuk, tehát visszafejtés nélkül nincs lehetőségünk belenézni az állományokba. A tárolt mappák jelentős része nem is látszik, kódolva tartalmazza őket a resources.arsc fájl a csomag gyökerében.
Az utolsó mondatokkal rá is tértünk az apk fájlok védelmére.
Mivel a nem nyílt forrású csomagok harmadik fél szellemi értékeit (az általa írt programot, grafikákat, stb.) tartalmazzák, meg kell őket védeni a rossz szándékú felhasználástól.
Ennek fényében a források nem, vagy csak részben visszafejthetőek, így nincs lehetőség a teljes forrás, vagy a forrás egyes részeinek máshol történő felhasználására.
Aki kibontott egy apk-t az bizonyára felfigyelt a META-INF mappára. Itt kerül tárolásra a csomag készítőjének aláírása, mert bizony minden egyes apk fájl egy egyedi kulccsal van hitelesítve.
Ez a kulcs szavatolja az apk védelmét a módosításokkal szemben.
Amennyiben egy bit is megváltozik, az apk-t futtató környezet észlelni fogja, hogy érvénytelenül aláírt elemek vannak a csomagban, és visszautasítja azt, tehát az apk nem lesz működőképes.
Megtehetjük azt, hogy generálunk saját kulcsot, amivel újra aláírjuk a csomagot, de az a csomag már nem a kiadó

saját hitelesítését fogja hordozni, így nyilvánvalóvá válik a felhasználó előtt, hogy egy nem hivatalos, nem eredeti kiadású csomagot használ.
Ezek a védelmi eljárások rányomhatnák a "FAIL" feliratú bélyeget a honosításokra, de szerencsére létezik egy lehetőség, amivel a módosított csomagokat használatba vehetjük a telefonon, amennyiben nem hivatalos romokra készítünk nem hivatalos update csomagokat.
Ahogy az egyes apk-knál, úgy a készülékek operációs rendszerét is védi egy kulcs, ennek eredményeként csak az operációs rendszer készítője által kiadott kulccsal aláírt frissítő csomagokat fogja elfogadni a recovery.
Itt jön képbe a root. Akinek nem probléma a garanciavesztés, megteheti, hogy feloldja a telefonján a külső behatások ellen védő eljárásokat, rendszergazda jogokkal férhet hozzá a telefonon futó busyboxhoz, vagy más recoveryt telepíthet a készülékre, vagy valamelyik harmadik féltől származó szoftvert installálhat a gyári rom helyére.
Innentől kezdve megkezdhetjük a kedvenc romunk magyarítását. Az első lépés, hogy a rom csomagjából, ami ugyancsak egy zip ki kell bontanunk az apk-kat, vissza kell fejtenünk, és lokalizálnunk őket.

Android honosítás 2. rész – Az apk csomagok kezelése
2010. május 18. | 2 hozzászólás | Kategória: Android OS, Bemutató, Fejlesztés

A sorozat előző részében megismerkedtünk az apk csomagok felépítésével, és most eljött az idő, hogy gyakorlatba ültessük az elméleti részt.
A fájlok kibontásával kell kezdenünk, amihez egy nagyon ügyes kis programot használunk, az apktoolt. A program ingyenesen letölthető innen.
Két fájlra lesz szükségünk, mégpedig magára az apktoolra, és a megfelelő platformra kiadott indító állományára.
Nem akarok sok időt vesztegetni, mivel az apktool telepítéséhez minden információ megvan az oldalukon. Aki elakadt, az nyugodtan kérdezzen.
Én Linux alapú operációs rendszert használok, és ajánlom mindenkinek, hogy akár virtuális gépben, de mindenképp valamilyen Linux alapú rendszer, esetleg Ubuntu alatt próbálgassa az itt leírtakat.
Természetesen Windows és MAC OS alatt is van lehetőség ezekre a dolgokra, ha valaki ragaszkodik ezekhez az operációs rendszerekhez.
Próbáljátok értelmezni a használt parancsokat, és alakítsátok őket a megfelelő rendszer parancssorához.
Miután már mindenünk megvan a csomagok kezeléséhez, bontsunk is ki egy APK fájlt. Lássuk hogy is működik az apktool:
$ apktool
Apktool v1.1.1 - a tool for reengineering Android apk files
Copyright 2010 Ryszard Wiśniewski
Usage: apktool [-v|--verbose] COMMAND [...]
COMMANDs are:
d[ecode] [-s|--no-src][-r|--no-res][-d|
Decode <file.apk> to <dir>.
-s, --no-src
Do not decode sources.
-r, --no-res
Do not decode resources.
-d, --debug
Decode in debug mode. Check project page for more info.
b[uild] [-f|--force-all][-d|--debug] [<app_path>]
Build an apk from already decoded application located in <app_path>.
Apk file will be placed in <app_path>/dist/out.apk.
It will automatically detect, whether files was changed and perform
needed steps only.
If you omit <app_path> then current directory will be used.
-f, --force-all
Skip changes detection and build all files.
-d, --debug
Build in debug mode. Check project page for more info.
For additional info, see: http://code.google.com/p/
Ha áttanulmányoztuk a súgót, bontsunk ki egy APK fájlt, legyen az a Launcher.apk:
$ apktool d -s Launcher.apk Launcher
I: Copying raw classes.dex file...
I: Decoding resource table...
I: Decoding resources...
I: Copying assets and libs...
Láthatjuk, hogy létrejött egy mappa Launcher néven. Ha belekukkantunk, a következő fájlokat fogjuk látni:
9patch AndroidManifest.xml classes.dex res
A kibontáshoz a “d”, avagy decode és a “-s”, azaz források nélkül kapcsolókat használtuk, így csak a res mappát bontottuk ki, a forrásfájlokhoz nem nyúltunk hozzá.
A kibontott/létrejött mappák a következők:
9patch. Az android csomagokban van lehetőségünk amolyan “kimaszkolt” képelemeket létrehozni.

Ez azért jó, mert az így létrehozott fájlokat a megfelelő paraméterekkel lehet nyújtani, nem szükséges a képek több méretbeni tárolása.
A 9patch mappában tárolt fájlok segítenek a maszkolt képek majdani visszaalakításában, a honosításokhoz nincs szükségünk rá.
AndroidManifest.xml. Az előző cikkben volt róla szó, lapozzatok vissza.
classes.dex.
Erről szintén az előző részben olvashattatok és, ahogy sanyiii említette, az src mappában tárolt Java forrásokból fordított bájtkód van benne (mégegyszer köszönöm neki a javítást).
res mappa. Ismét az előző részben került említésre. Nézegessétek, tanulmányozzátok az itt tárolt fájlokat, melyeket a következő, befejező cikkben részletesen is kitárgyalunk.
Térjünk vissza a példánkhoz, ahol ugye már kicsomagoltuk az első APK csomagunkat.
A kísérletezni vágyó olvasók csinálhatnak rajta módosításokat, viszont most lássuk, hogy hogyan is lehet újra APK-t varázsolni a könyvtárakból:

```
$ apktool b Launcher
I: Copying classes.dex file...
I: Checking whether resources has changed...
I: Building resources...
I: Building apk file...
```

A classes.dex úgy, ahogy van be lett másolva a csomagba, ellenőrizve lettek a módosítások, összepakoltuk a res mappát, és létrehoztuk az apk csomagot.
Most a Launcher könyvtárban létrejött egy “dist” mappa, amiben megtalálhatjuk az out.apk fájlt. Ez a most összecsomagolt APK fájlunk, ami viszont még nincs aláírva. Az aláíráshoz először kulcsot kell gyártanunk magunknak. A kulcs létrehozásához telepítve kell legyen a Java JDK a gépünkre. Windows felhasználók innen tölthetik le. Linux felhasználók csak telepítsék repoból (ajánlom az openjdk-t).
Amennyiben sikerült feltelepíteni a JDK-t, generáljuk le a kulcsunkat:

```
keytool -genkey -alias kulcsom.keystore -keyalg RSA -validity 20000 -keystore kulcsom.keystore
```

Sikeresen létrehoztuk a saját privát kulcsunkat, amivel aláírhatjuk az APK csomagot:

```
jarsigner -verbose -keystore kulcsom.keystore -signedjar Launcher/dist/Launcher.apk Launcher/dist/out.apk
kulcsom.keystore
```

Most létrejött a Launcher/dist/Launcher.apk, ami már az újonnan lefordított APK fájlunk aláírt verziója.

Android honosítás 3. rész – Honosítás és Update csomag
2010. június 3. | 33 hozzászólás | Kategória: Android OS, Bemutató, Fejlesztés

Elég sok idő telt el a második rész óta, remélhetőleg minden érdeklődő ki tudta próbálni az korábbi részekben leírt dolgokat.
Tehát van néhány kibontott apk fájlunk, egy telefonunk, ami most magyarítani szeretnénk. Elöljáróban néhány szót a res mappa működéséről.
A leggyakoribb könyvtárak, amiket itt találhatunk a következők.
drawable. Itt a programcsomag által használt grafikai elemek, és néhány xml fájl található, melyek a grafikus elemek elhelyezkedését/viselkedését befolyásolják.
layout. Az itt található xml fájlok szabják meg az alkalmazás ablakainak/képernyőinek/
values. Számunkra ez igen fontos, hiszen többek között a nyelvi fájlokat is tárolja ez a mappa, amibe később részletesebben is belenézünk.
xml. A program által használt leíró fájlok, amelyek különböző adatokat tartalmazhatnak és a program működését befolyásolják.
Észrevehettétek, hogy néhány mappából több változat is létezik, ami számunkra nagyon fontos.
Különböző nyelveknek, telefon típusoknak külön mappát tudunk készíteni. Hogy is működik ez?
Példának vegyük az 1.6-os Launcher.apk-t.
Ebben a csomagban van egy drawable könyvtár, amiben a program képeit találjuk, valamint van egy drawable-land és egy drawable-port mappa.
Attól függően, hogy a telefon álló vagy fekvő módban van, a szükséges elemeket a drawable-land és drawable-port mappákból fogja venni.
Hasonlóan ehhez, a values mappában az alapértelmezett angol fájlokat találjuk, például a values-it az olasz nyelvi fájlokat hordozza.
Ha létrehoznánk egy values-it-land mappát, akkor a fekvő módhoz tartozó olasz szövegeket innen töltené be.
Ezen a logikán tovább haladva, hozzunk létre egy values-hu mappát, amivel hozzá is foghatunk a magyarításhoz.
A values mappában több xml fájlban van, de nekünk nem lesz mindre szükségünk.
Ha megtekintjük a values-it mappát, akkor könnyedén lemásolhatjuk, hogy vajon mely fájlokra van szükségünk a magyar lokalizáció létrehozásához.
Másoljuk át a strings.xml-t a values mappából a values-hu mappába. A nyelvi fájlok az Androidtól megszokott xml formátumban vannak tárolva.
Windows felhasználók, figyelem! A fájlokat UTF-8 kódolással kell menteni, különben nem fognak működni az ékezetes karakterek.
Windows alatt ajánlom az ingyen letölthető notepad2 alkalmazást. Én Linux rendszeren a geditet használom.
Nyissuk meg a values-hu mappából a strings.xml-t, és nézzük meg a szerkezetét.

<?xml version="1.0" encoding="UTF-8"?>
<resources>
<string name="application_name">Home</
<string name="uid_name">Android Core Apps</string>
<string name="folder_name">Folder</
<string name="chooser_wallpaper">
...
</resources>
A stringek felépítése egyszerű: <string name="string_neve">Szöveg</.
Ezek alapján fordítsuk le a fenti angol szöveget:
<?xml version="1.0" encoding="UTF-8"?>
<resources>
<string name="application_name">
<string name="uid_name">Android Alkalmazások</string>
<string name="folder_name">Mappa</
<string name="chooser_wallpaper">
...
</resources>
Ennyi lenne.
Nem kell patchelni a forráskódot, hogy elhelyezzük a magyarítást, nem kell beírogatni a magyar stringeket a kódba.
Egyszerűen átmásoljuk az angol fájlokat a -hu végű mappákba, és lefordítjuk őket. Van azért néhány dolog, amire figyelni kell.
A "\" karakter a backspace. Az Android rendszer érzékeny a különböző különleges karakterekre, mert sokat operátorként használ fel (", ~, ', stb.)
Ezek előtt a karakterek előtt "\" jelet szokás kitenni, így értesítve a rendszert, hogy a következő karakter egy írásjel lesz.
Sok helyen fordul elő változó, ami valamilyen adattal, vagy szöveggel lesz felcserélve:
Shortcut \"%s\" created… Itt látszik, hogy az idézőjelek backspace-vel vannak ellátva, a "%s" pedig egy változó, ahova a rendszer az ikon nevét fogja behelyettesíteni.
Ilyenkor érdemes végiggondolni, hogy nyelvtanilag mennyire lenne helyes a mondat, ha a változót ott hagynánk ahol van, pl.: Ikon "Gmail" létrehozva. vagy inkább "Gmail" ikon létrehozva.
Nagyon figyeljünk oda, hogy a string nevét ne írjuk át, ne töröljük ki a <> jeleket.
Amennyiben kész a magyarítás, az előzőekben ismertetett módon fordítsuk újra, írjuk alá, és hozzuk létre az update csomagot.
Maga az update csomag egy .zip fájl, és hasonlóan kell kinéznie, mint a ROM-nak, amiből a csomag származik.
Az update csomagnak van egy fontos könyvtára, a META-INF/com/google/android, ahol találunk egy update-script nevű fájlt.
Ebben van meghatározva, hogy hogyan történjen az update.
Ezt mindig az adott ROM update scriptjéből érdemes értelemszerűen kibogarászni, de a következő parancsok általában működnek.

show_progress 0.5 0

copy_dir PACKAGE:system SYSTEM:

show_progress 0.5 10

A fentiekből látszik, hogy az update folyamat a system könyvtárat fogja felülírni az általunk mellékelt fájlokkal.
Ezek után másoljuk be az apk fájlunkat a helyére. (A pontos helyét nézzük meg az eredeti update csomagban, amiből kivettük.)
Ez nálam a system/app/Launcher.apk. Majd a kedvenc archívumkezelőnkkel készítsük el a zip csomagot.
Szinte készen vagyunk, már csak az aláírás szükséges.
Sajnos az apk csomagoktól eltérő metódust kell alkalmaznunk, ebben segítséget találunk ezen a hivatkozáson.
Innentől kezdve már csak újra kell indítanunk a telefont Recovery módban, csinálni egy backupot, és flashelni a kapott fájlt.
Sok sikert kívánok a magyarításhoz, és bombázzatok a kérdéseitekkel!
Szerző: Apoth