POGLAVLJE 1

Početak

1.1 Uvod: početak

Ian Darwin

Objašnjenje

Čuveni primer „Hello, World“ (zdravo svete) nastao je kada su Kernighan i Plaugher poželeli da napišu „recept“ za početnike u svakom novom programskom jeziku i okruženju. Ovo poglavlje je posvećeno njima i svakome ko se mučio da savlada nov programski jezik.

1.2 Učenje jezika Java

Ian Darwin

Problem

Android aplikacije se pišu na programskom jeziku Java pre nego što se prevedu u Androidov format datoteka klasa, DEX. Ako ne znate da programirate na Javi biće vam teško da pišete Android aplikacije.

Rešenje

Na raspolaganju je više izvora za učenje Jave. Većina njih će vas naučiti onome što vam je potrebno ali će vam predstaviti i neke API klase koje nisu na raspolaganju za razvijanje Android aplikacija. U svim izvorima informacija *izbegavajte* delove koji govore o temama iz leve kolone tabele 1-1.

Tabela 1-1. Delovi Javinog okruženja za programiranje koje treba zanemariti

Javin API	Ekvivalent u Androidu
Swing, apleti	Androidov grafički korisnički interfejs (videti poglavlje 7)
Metoda `main()` kao ulazna tačka aplikacije	Videti Recept 1.6
J2ME/JavaME	Najveći deo paketa `android.*` zamenjuje Java ME API
Servleti/JSP, J2EE/Java EE	Namenjeno za upotrebu na serverima

Objašnjenje

Evo nekih knjiga i izvora informacija o programiranju na jeziku Java:

- *Java in a Nutshell* (*http://shop.oreilly.com/product/9780596007737.do*), čiji je autor David Flanagan (izdavač je O'Reilly), dobar je uvod za programere, posebno za one koji prelaze s jezika C/C++. Obim ove knjige se povećavao uporedo s razvojem jezika Java SE.
- *Head First Java* (*http://shop.oreilly.com/product/9780596009205.do*), čiji su autori Kathy Sierra i Bert Bates (izdavač je O'Reilly) pruža odličan uvod u Javu, namenjen onima kojima je lakše da vizuelno shvate materiju.
- *Misliti na Javi*, prevod 4. izdanja (Mikro knjiga, 2007), Brus Ekel, (izdavač originala Prentice-Hall).
- *Learning Java* (*http://shop.oreilly.com/product/9780596008734.do*), Patrick Niemeyer i Jonathan Knudsen (O'Reilly).
- „Great Java: Level 1" (*http://shop.oreilly.com/product/9780596009393.do*), video-zapis čiji je autor Bret McLoughlin (O'Reilly), pruža vizuelan uvod u Javu.
- *Java: The Good Parts* (*http://shop.oreilly.com/product/9780596803742.do*), Jim Waldo (O'Reilly).
- *Java Cookbook* (*http://shop.oreilly.com/product/9780596007010.do*) čiji sam autor ja, a izdavač je O'Reilly, smatra se dobrom pomoćnom knjigom za programere na Javi. Cela poglavlja su posvećena nizovima, regularnim izrazima, brojevima, datumima i vremenima, strukturiranju podataka, ulazno-izlaznim operacijama s podacima i direktorijumima, internacionalizovanju, višenitnom radu i umrežavanju, a sve to je primenljivo i na Android. Knjiga sadrži i nekoliko poglavlja koja su specifična za Swing, kao i za neke tehnologije specifične za Javino izdanje EE.
- *Java JDK 7: kompletan priručnik*, prevod 8. izdanja (Mikro knjiga, 2012), Herbert Schildt (izdavač orginala McGraw-Hill).

Imajte u vidu to da ova lista verovatno nikada neće biti sasvim ažurna. Trebalo bi da pogledate i knjigu *Android Development Bibliography* (*http://shop.oreilly.com/product/0636920021896.do*) koja se preuzima besplatno (posle registrovanja) sa veb lokacije izdavačke kuće O'Reilly, a predstavlja kompilaciju svih knjiga raznih izdavača čije su knjige dostupne preko internet servisa Safari. Ta knjiga se takođe besplatno deli na relevantnim konferencijama gde O'Reilly ima svoj štand.

Videti i

Glavni autor ove knjige održava i listu izvora informacija o Javi na adresi *http://www.darwinsys.com/java/*.

O'Reilly je objavio neke od najboljih knjiga o Javi: kompletnu listu izdanja pogledajte na adresi *http://oreilly.com/pub/topic/java*.

1.3 Izrada aplikacije „Hello, World“ sa komandne linije

Ian Darwin

Problem

Želite da napravite nov Android projekat a da pri tome ne koristite razvojno okruženje Eclipse i dodatni modul ADT.

Rešenje

Upotrebite alatku `android` iz razvojnog kompleta Android Development Kit (ADK) kojoj zadate argument `create project` i neke dodatne argumente pomoću kojih konfigurišete projekat.

Objašnjenje

Osim što je ime platforme, `android` je i naziv alatke (naredbe) za pravljenje, ažuriranje i upravljanje projektima s komandne linije. Možete preći u direktorijum *android-sdk-xxx* ili podesiti sistemsku promenljivu `PATH` tako da sadrži poddirektorijum *tools* tog direktorijuma.

Zatim, da biste napravili nov projekat, zadajte komandu `android create project` s odgovarajućim argumentima. Primer 1-1 prikazuje izvršavanje te komande u prozoru Command Prompt.

Primer 1-1. Izrada novog projekta

```
C:> PATH=%PATH%;"C:\Documents and Settings\Ian\My Documents\android-sdk-windows\
   tools"; \"C:\Documents and Settings\Ian\My Documents\android-sdk-windows\
             platform-tools"
C:> android create project --target 1 --package com.example.foo
     --name Foo --activity FooActivity --path .\MyAndroid
Created project directory: C:\Documents and Settings\Ian\My Documents\MyAndroid
Created directory C:\Documents and Settings\Ian\My Documents\MyAndroid\src\com\
   example\foo
Added file C:\Documents and Settings\Ian\My Documents\MyAndroid\src\com\example\
   foo\FooActivity.java
Created directory C:\Documents and Settings\Ian\My Documents\MyAndroid\res
Created directory C:\Documents and Settings\Ian\My Documents\MyAndroid\bin
Created directory C:\Documents and Settings\Ian\My Documents\MyAndroid\libs
Created directory C:\Documents and Settings\Ian\My Documents\MyAndroid\res\values
Added file C:\Documents and Settings\Ian\My Documents\MyAndroid\res\values\
   strings.xml
Created directory C:\Documents and Settings\Ian\My Documents\MyAndroid\res\layout
Added file C:\Documents and Settings\Ian\My Documents\MyAndroid\res\layout\
   main.xml
Added file C:\Documents and Settings\Ian\My Documents\MyAndroid\
   AndroidManifest.xml
Added file C:\Documents and Settings\Ian\My Documents\MyAndroid\build.xml

C:>
```

Sledi lista argumenata za program `create project`.

Tabela 1-2. Lista argumenata komande create project

Ime	Značenje	Primer
`--activity`	Naziv vaše „glavne klase" i podrazumevano ime za generisanu *.apk* datoteku.	`--activity HelloActivity`
`--name`	Ime projekta i generisane *.apk* datoteke.	`--name MyProject`
`--package`	Ime Java paketa u kojem se nalaze vaše klase.	`--package com.example.hello`
`--path`	Putanja u kojoj se pravi projekat (pošto se ne prave poddirektorijumi unutar nje, nemojte zadavati samo `/home/`*`ime korisnika`*`/neki_direktorijum`, nego `/home/`*`ime korisnika`*`/neki_direktorijum/`*`ImeNovogProjekta`*).	`--path /home/ian/workspace/MyProject` (pogledajte gornji primer za Windows)
`--target`	Ciljni API nivo Android platforme; zadajte komandu `android list targets` da biste dobili listu mogućih ciljnih platformi. Broj koji dobijete je „ID", a ne API nivo; taj identifikator se zadaje u komandi `android` – API nivo koji želite.	`--target android -10`

Ukoliko ne može da obavi traženu operaciju, komanda `android` prikazuje obiman listing s primerima upotrebe komande za sve operacije koje može da obavi i argumentima za njih. Ako uspešno obavi traženu operaciju, komanda `android create project` pravi sledeće datoteke i direktorijume.

Tabela 1-3. Elementi koje pravi komanda create project

Naziv	Značenje
AndroidManifest.xml	Konfiguraciona datoteka koja saopštava Androidu informacije o projektu
bin	Generisane binarne datoteke (prevedene datoteke klasa)
build.properties	Datoteka svojstava koja se može naknadno ažurirati
build.xml	Standardna Ant kontrolna datoteka
default.properties ili *project.properties* (u zavisnosti od verzije alatke)	Sadrži verziju SDK kompleta i biblioteka koje se koriste u projektu; održava je verzija dodatka ADT
gen	Generisani materijal
libs	Biblioteke, naravno
res	Važne datoteke resursa (`strings.xml`, datoteke prikaza itd.)
src	Izvorni kôd aplikacije
src/imepaketa/ImeAktivnosti.java	Izvorni kôd vaše „glavne" početne aktivnosti
test	Kopije većine prethodnih elemenata

Uobičajena i preporučena praksa u Androidu jeste da korisnički interfejs aplikacije definišete u obliku XML datoteke koju ćete smestiti u direktorijum `res/layout`, ali je svakako moguće i napisati ceo kôd na Javi. Da bi ovaj primer bio samostalna celina, zasad ćemo ga

napisati na „pogrešan“ način. U svom omiljenom editoru teksta zamenite sadržaj datoteke HelloWorld.java sadržajem iz primera 1-2.

Primer 1-2. HelloWorld.java

```
import android.app.Activity;
import android.widget.*;

public class HelloWorld extends Activity {

        /**
        * Ova metoda se poziva kada se aktivnost incijalizuje, kao rezultat
        * određene akcije, npr. biranja ikonice aplikacije na ekranu Home.
        */
        @Override
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                // Izrada polja tipa TextView u tekućoj aktivnosti
                TextView view = new TextView(this);
                // Neka polje sadrži nešto
                view.setText("Hello World");
                // Novo polje povezujemo sa aktivnošću,
                // otprilike kao JFrame.getContentPane().add(view)
                setContentView(view);
        }
}
```

Ako pretpostavimo da ste instalirali alatku Ant Build fondacije Apache Software (*http://ant.apache.org*), (sadrže je novije verzije razvojnog kompleta Android SDK), sada možete (u prozoru Command Prompt) preći u direktorijum projekta (`...MyDocuments\MyAndroid` u primeru 1-1) i zadati komandu:

```
ant debug
```

Time ćete napraviti arhivu sa imenom npr. *MyAndroid.apk* („apk“ je skraćenica za Android Package) u direktorijumu `bin`.

Ako ovo radite po prvi put, možda ćete morati da napravite *virtuelni Android uređaj* (engl. *Android Virtual Device*, *AVD*), što je imenovana konfiguracija za Androidov emulator u kojoj zadajete ciljnu rezoluciju ekrana, API nivo itd. Emulator možete napraviti pomoću sledeće komande:

```
android create avd -n my_droid -t 7
```

Više detalja o izradi AVD-a dato je u receptu 3.3.

Zatim možete pokrenuti ADB (Android Debug Bridge) server i emulator:

```
adb start-server
emulator -avd my_droid -t 5
```

Pod pretpostavkom da je pokrenut emulator ili da je vaš fizički uređaj povezan i prepoznat preko USB priključka, možete zadati komandu:

```
adb -e install -r bin/MyAndroid.apk
```

Zadajte indikator `-e` ako je u pitanju emulator, odnosno `-d`, ukoliko se radi o stvarnom uređaju.

Ako se snalazite s komandnim skriptovima ili komandnim datotekama, verovatno ćete napraviti datoteku koja se zove, recimo, *download* kako biste izbegli pozivanje komande `adb` u svakom ciklusu pisanja programa.

Konačno možete da pokrenete svoju aplikaciju! Možete upotrebiti listu aplikacija: pritisnite ikonicu koja izgleda kao mreža 5 × 5 tačaka, izlistajte sadržaj dok ne dođete do imena svoje aplikacije i pritisnite njenu ikonicu.

Možda će vam biti pogodnije da ikonicu aplikacije postavite na početni ekran uređaja ili emulatora; pošto će tako ikonica preživeti višekratna ponavljanja komande `install -r`, to je najlakši način da testirate aplikaciju.

Videti i

Recept 1.4. Blog „a little madness“ sadrži detaljnije informacije (*http://www.alittlemadness.com/2010/05/31/setting-up-an-android-project-build/*). Zvanična referentna Androidova veb lokacija ima stranicu o programiranju bez razvojnog okruženja Eclipse (*http://developer.android.com/guide/developing/other-ide.html*).

1.4 Izrada aplikacije „Hello, World“ u razvojnom okruženju Eclipse

Ian Darwin

Problem

Želite da koristite razvojno okruženje Eclipse kako biste napravili Android aplikaciju.

Rešenje

Instalirajte razvojno okruženje Eclipse (*http://www.eclipse.org/*), razvojni komplet Android SDK (*http://developer.android.com/sdk/*) i dodatni modul ADT (*http://developer.android.com/sdk/eclipse-adt.html*). Otvorite nov projekat i počnite da pišete aplikaciju. Kad završite, prevedite je i testirajte je na emulatoru, u okruženju Eclipse.

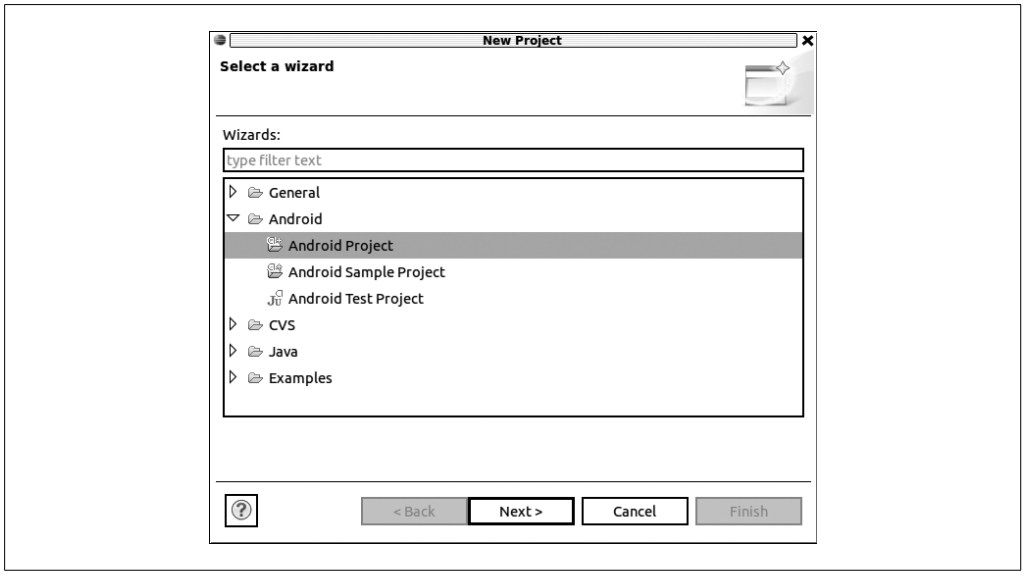
Objašnjenje

Pošto instalirate sledeće elemente, spremni ste da počnete:

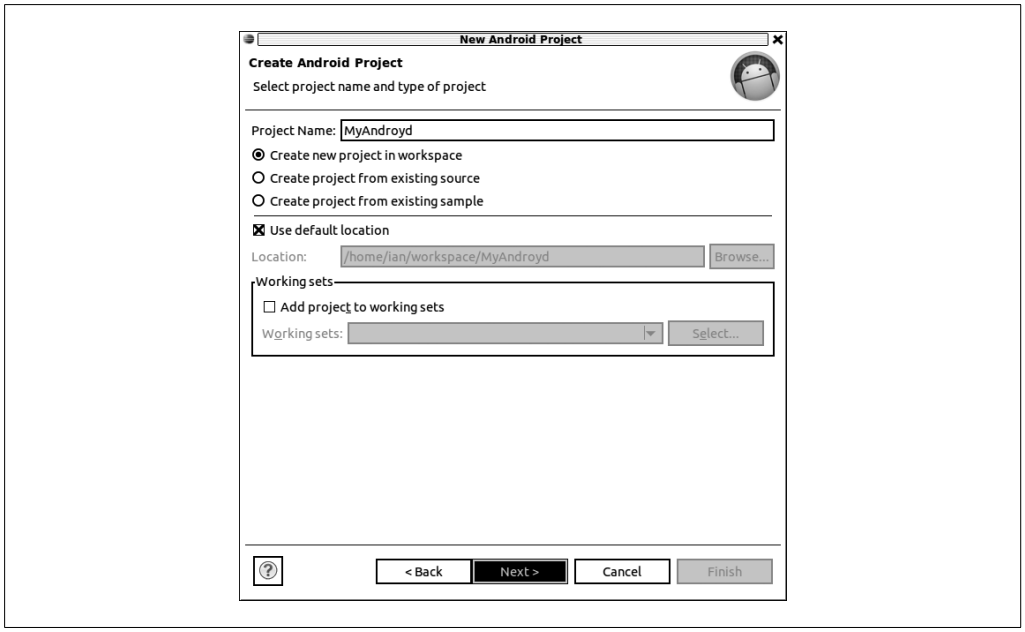
- Integrisano razvojno okruženje Eclipse (*http://www.eclipse.org/*)
- Razvojni komplet Android SDK (*http://developer.android.com/sdk/*)
- Dodatni modul za Eclipse, ADT (*http://developer.android.com/sdk/eclipse-adt.html*)

Ako vam trebaju detaljnija objašnjenja o instaliranju navedenih elemenata, pogledajte recept 1.5.

Da biste počeli s radom, otvorite nov projekat pomoću opcije menija File→New (slika 1-1). Pritisnite Next. Zadajte ime novog projekta i pritisnite Next (slika 1-2).



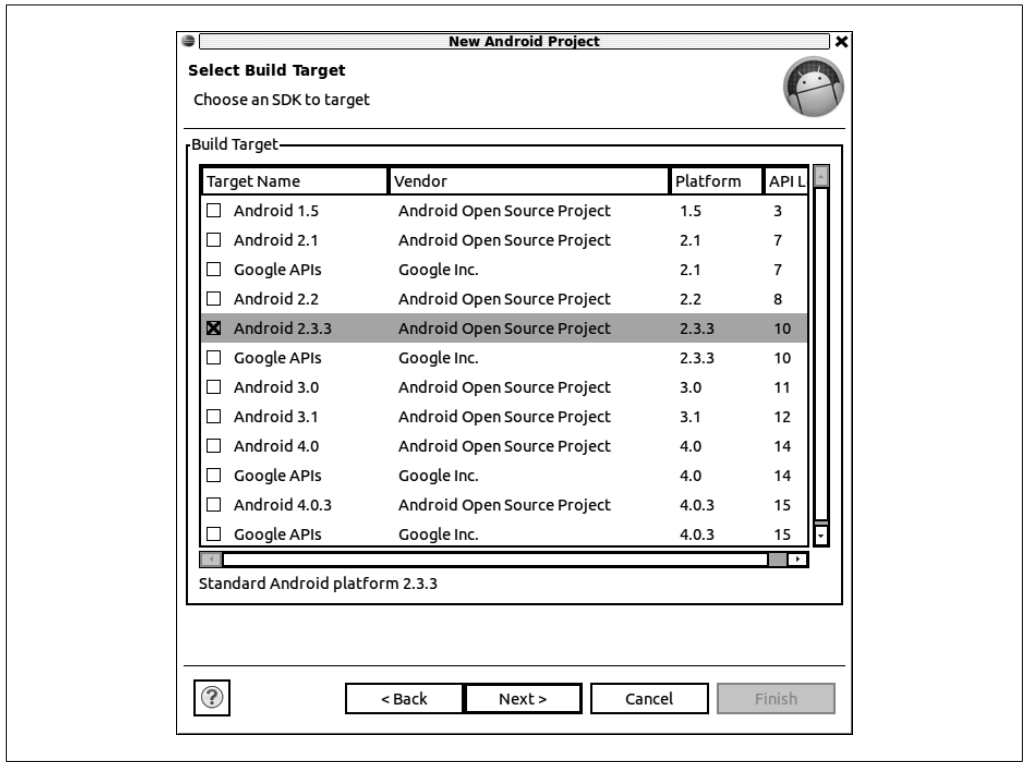
Slika 1-1. Započinjanje novog projekta u razvojnom okruženju Eclipse



Slika 1-2. Zadavanje parametara za nov projekat u okruženju Eclipse

Izaberite ciljnu verziju SDK za Android. Verzija 2.1 sadrži gotovo sve uređaje koji se danas koriste; verzije 3.x i 4.x pružaju najnovije mogućnosti (slika 1-3). Odlučite sami.

Slika 1-4 prikazuje strukturu projekta u oknu Project s leve strane. Ona takođe pokazuje do kog nivoa možete koristiti opciju Autocomplete iz Eclipsea u Androidu. Natpisu sam dodao atribut `gravity`, a Eclipse mi je ponudio kompletnu listu mogućih vrednosti tog atributa. Izabrao sam vrednost `center-horizontal`, pa bi trebalo da natpis bude centriran kada aplikacija radi.



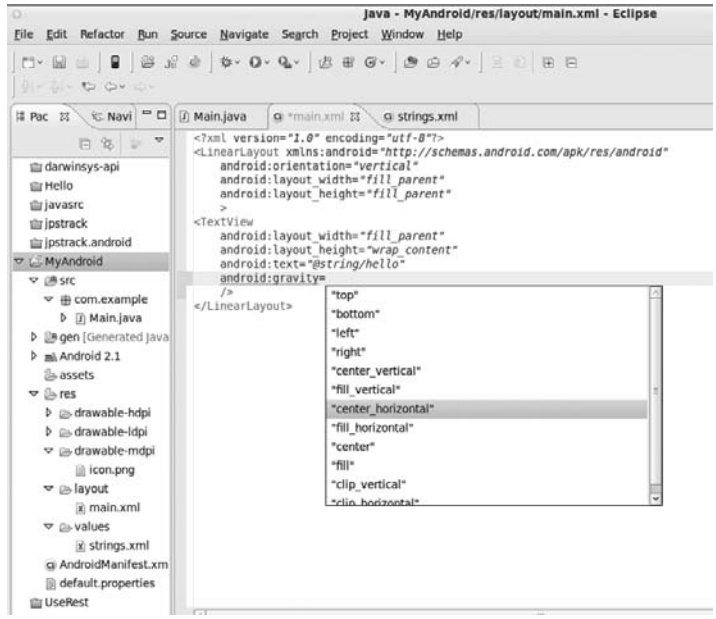
Slika 1-3. Zadavanje ciljne verzije SDK za nov projekat u okruženju Eclipse

Zapravo, ako za ceo ekran zadate raspored elemenata `LinearLayout` *i* atribut `gravity` polja tipa `TextView` podesite na `center_horizontal`, tekst u polju biće centriran i horizontalno i vertikalno. Primer 1-3 je datoteka rasporeda elemenata (engl. *layout file*) *main.xml* (nalazi se u direktorijumu *res/layout*) kojom se ovo postiže.

Primer 1-3. XML datoteka sadržaja ekrana

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
```

```
    android:layout_height="fill_parent"
    android:gravity="center_vertical"
    >
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello"
    android:gravity="center_horizontal"
    />
</LinearLayout>
```



Slika 1-4. Zadavanje vrednosti atributa gravity pomoću editora u razvojnom okruženju Eclipse

Kao i uvek, Eclipse generiše prevedenu (engl. *compiled*) verziju kad god izvornu datoteku snimite na disk. U Android projektu takođe pokreće alatku Ant Build da bi napravio prevedenu i zapakovanu APK datoteku koja je spremna za rad. Treba samo da je pokrenete. Desnim tasterom miša pritisnite projekat i izaberite opciju Run As → Android Project (slika 1-5).

To će pokrenuti Androidov emulator, ako još nije pokrenut. Na početnom ekranu emulatora biće ispisana reč *Android* fontom koji podseća na slova pisaće mašine a zatim će se prebaciti na uglađeniji Android font s pokretnom belom kockicom preko plavih slova – sećate se pokretanja Microsoftovog Windowsa 95? Videti sliku 1-6.

Slika 1-5. Pokretanje Android projekta u okruženju Eclipse

Slika 1-6. Pokretanje Android projekta na emulatoru

Koji trenutak kasnije, trebalo bi da se pokrene vaša aplikacija (slika 1-5 prikazuje samo snimak ekrana aplikacije budući da nam ostatak prikaza u emulatoru nije bitan). Videti sliku 1-7.

Videti i

Recept 1.3

Slika 1-7. Projekat iz okruženja Eclipse dok radi na emulatoru

1.5 Podešavanje integrisanog razvojnog okruženja u Windowsu za razvoj Android aplikacija

Daniel Fowler

Problem

Pošto želite da pravite Android aplikacije na PC računaru koji radi pod Windowsom, koristiće vam sažet vodič za podešavanje integrisanog razvojnog okruženja za tu platformu.

Rešenje

Za razvoj aplikacija za Android preporučuje se upotreba integrisanog razvojnog okruženja Eclipse. Konfigurisanje okruženja Eclipse u Windowsu nije jednostavno budući da se postupak sastoji od više koraka. Ovaj recept objašnjava te korake.

Objašnjenje

Za razvijanje aplikacija za Android preporučuje se upotreba integrisanog razvojnog okruženja Eclipse za Javu. Za to okruženje postoji dodatni modul Android Development Tools (ADT) koji koristi razvojni komplet Android Software Development Kit (SDK) gde se nalaze glavne alatke za razvoj softvera za Android. Da biste podesili razvojni sistem, treba da preuzmete i instalirate sledeće komponente:

- Java Standard Edition Development Kit
- Eclipse for Java Development
- Android Software Development Kit
- Dodatni modul Android Development Tools (iz okruženja Eclipse)

U nastavku slede detaljniji opisi navedenih koraka za PC računar s Windowsom (testirano na XP-u, Visti i Windowsu 7).

Instaliranje razvojnog kompleta JDK (Java Development Kit)

Pređite na veb stranicu za preuzimanje Java softvera (*http://www.oracle.com/technetwork/java/javase/downloads/index.html*)

Izaberite ikonicu Java da biste preuzeli JDK:

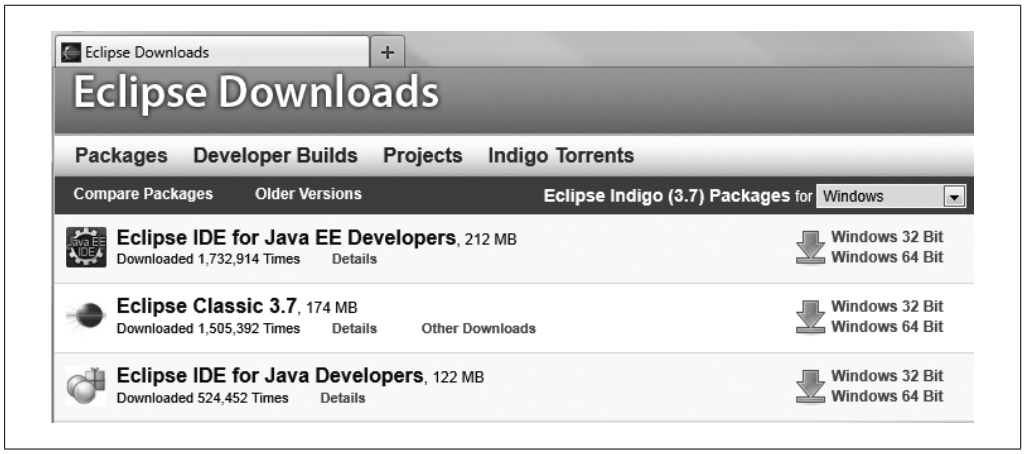
Prikazaće se lista za preuzimanje JDK. Pritisnite radio-dugme „Accept License Agreement“ jer u suprotnom nećete moći da preuzmete JDK. Preuzmite i pokrenite najnoviju verziju JDK koju nađete; u vreme pisanja ove knjige to je bila datoteka *jdk-7-windows-i586.exe* (ili *jdk-7-windows-x64.exe* za 64-bitnu verziju Windowsa). Možda ćete morati da odaberete veb lokaciju za preuzimanje. Prihvatite sva bezbednosna upozorenja koja se pojave ali samo ako preuzimate softver sa zvanične Javine veb lokacije za preuzimanje.

Pošto preuzmete datoteku i pokrenete je, moraćete da prođete kroz nekoliko instalacionih ekrana – pritiskajte dugme Next sve dok instalacioni program za JDK ne završi svoj posao. Ne bi trebalo da menjate nijednu ponuđenu opciju. Kada instalacioni program za JDK završi, pritisnite dugme Finish. Možda će se prikazati veb stranica za registraciju softvera; možete je zatvoriti ili odabrati da registrujete softver koji ste instalirali.

Instaliranje okruženja Eclipse za razvijanje Java programa

Pređite na web stranicu za preuzimanje razvojnog okruženja Eclipse (*http://www.eclipse.org/downloads/*)

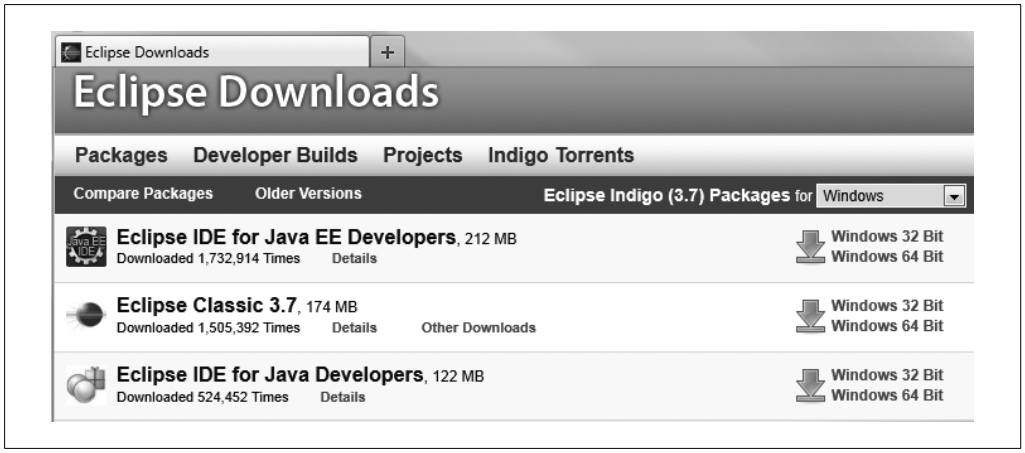
Na padajućoj listi Packages treba da bude izabrana opcija Windows; izaberite odgovarajuću hipervezu Eclipse IDE for Java Developers (slika 1-8).



Slika 1-8. Izbor razvojnog okruženja Eclipse za preuzimanje

Preuzmite i raspakujte .zip arhivu. U toj arhivi se nalazi direktorijum *eclipse* koji sadrži više datoteka i poddirektorijuma. Kopirajte direktorijum *eclipse* i ceo njegov sadržaj (slika 1-9). Uobičajeno mesto za kopiranje datoteka je korenski direktorijum na disku C ili direktorijum *C:\Program Files*; možda ćete morati da izaberete Continue kada Windows zatraži dozvolu za kopiranje.

Na radnoj površini (engl. *desktop*) napravite prečicu za datoteku *eclipse.exe*.



Slika 1-9. Sadržaj direktorijuma Eclipse

Pokrenite Eclipse da bi se konfigurisao direktorijum za radni materijal (engl. *workspace*); pri tome se proverava i jesu li Java i Eclipse pravilno instalirani. Pošto pokrenete okruženje Eclipse, može se pojaviti bezbednosno upozorenje; prisnite Run da biste nastavili dalje. Prihvatite ponuđeni direktorijum za radni materijal ili zadajte drugi direktorijum.

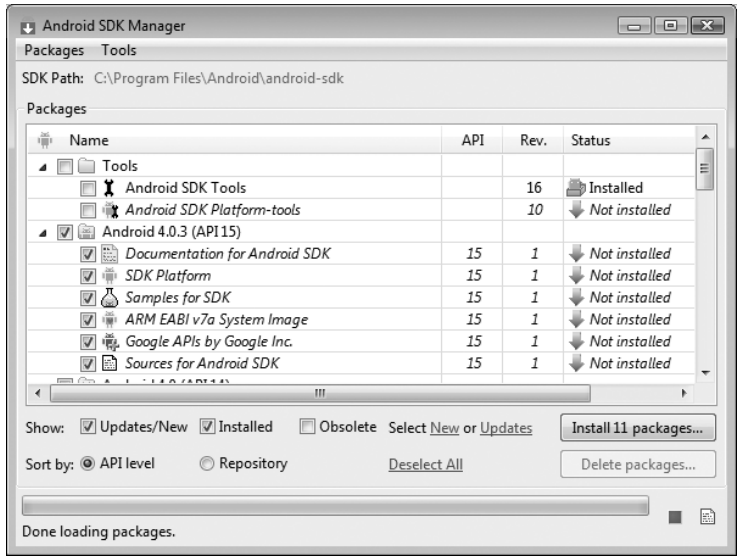
Instaliranje razvojnog kompleta Android SDK (Software Development Kit)

Pređite na veb lokaciju za preuzimanje razvojnog kompleta Android Software Development Kit (*http://developer.com/sdk/index.html*).

Izaberite najnoviji paket Windows EXE (zasad je to *installer_r16-windows.exe*) i pritisnite Run. Prihvatite bezbednosno upozorenje jedino ako preuzimate softver sa zvanične veb lokacije za preuzimanje razvojnog kompleta Android SDK. Program za instaliranje razvojnog kompleta Android SDK prikazaće nekoliko ekrana; u svakom pritisnite dugme Next; ne bi trebalo da menjate nijednu ponuđenu opciju. Pošto je *C:\Program Files* zaštićen direktorijum, pribavite ovlašćenje za instaliranje u taj direktorijum ili softver instalirajte u svoj korisnički direktorijum ili neki drugi, na primer u *C:\Android\android-sdk*.

Kad pritisnete dugme Install, nakratko će se prikazati ekran na kome možete pratiti napredovanje postupka instaliranja dok se Android datoteke kopiraju. Pritisnite poslednje dugme Next i dugme Finish na kraju postupka instalacije. Ako ste ostavili znak potvrde u polju Start SDK Manager, pokrenuće se alatka SDK Manager. U suprotnom, iz grupe programa Android SDK Tools izaberite SDK Manager (Start→All Programs→Android SDK Tools→SDK Manager). Kada se pokrene, SDK Manager proverava koji su Android paketi na raspolaganju za preuzimanje. Prikazaće se lista dostupnih paketa na kojoj su neki unapred izabrani za preuzimanje. Kolona Status pokazuje da li je određeni paket već instaliran. Na slici 1-10 vidi se da je paket Android SDK Tools upravo instaliran, što odražava i kolona Status.

Utvrdite koje sve pakete treba da istalirate. Na raspolaganju je više njih. Među njima su SDK paketi specifični za platformu i za svaki API nivo, primeri aplikacija za većinu API nivoa, API-ji za Google Maps, API-ji specifični za uređaje određenih proizvođača, dokumentacija, izvorni kôd i sledeći dodatni Googleovi paketi:



Slika 1-10. Androidov SDK Manager, koji prikazuje već instalirane komponente i one koje treba preuzeti

Android Support

Podrška za najnovije API-je na starijim uređajima

AdMob Ads SDK

Omogućava ugrađivanje reklama u aplikacije

Analytics SDK

Podržava analizu kupovina koje je kupac obavio

Market Billing

Dodaje podršku za kupovanje iz aplikacija

Market Licensing

Omogućava zaštitu aplikacija od nelegalnog kopiranja

USB Driver

Za otklanjanje grešaka na fizičkim uređajima (ili pomoću upravljačkog programa proizvođača uređaja)

Webdriver

Omogućava testiranje kompatibilnosti veb lokacije s Androidovim čitačem veba.

Preporučljivo je da preuzmete više raznih SDK platformi kako biste mogli da testirate aplikaciju na više različitih konfiguracija uređaja. Vredi napomenuti da će stariji računari imati teškoća sa izvršavanjem emulatora uređaja za najnovije nivoe Androidovih API-ja; iz tog razloga je bolje da na takvim računarima razvijate aplikacije namenjene starijim SDK platformama. Ako niste sigurni šta bi sve trebalo da preuzmete, prihvatite ponuđene opcije ili ponovo pokrenite SDK Manager da biste druge pakete preuzeli kad i ako vam zatrebaju; ili potvrdite sve pakete kako biste ih sve preuzeli (to će možda malo potrajati). Pritisnite dugme „Install packages".

Prikazaće se lista izabranih paketa; paket za koji postoje uslovi licence koji zahtevaju prihvatanje, obeležen je znakom pitanja. Istaknite svaki paket obeležen znakom pitanja da biste pročitali uslove licence. Pomoću radio-dugmadi možete prihvatiti ili odbaciti paket. Odbačeni paketi se obeležavaju simbolom × crvene boje. Druga mogućnost je da izaberete opciju Accept All kako biste preuzeli sve što je na raspolaganju. Pritisnite dugme Install; prikazaće se dnevnik napredovanja s paketima u toku instaliranja, kao i greškama koje pri tome nastanu. Na Windowsu je česta greška da SDK Manager ne uspeva da nađe ili preimenuje određene direktorijume. Ponovo pokrenite SDK Manager kao administrator i ispitajte da li su neki direktorijum ili datoteka podešeni samo za čitanje; više detalja o tome naći ćete u receptu 1.12. Kada se postupak završi, zatvorite SDK Manager tako što ćete pritisnuti dugme × u gornjem desnom uglu njegovog prozora.

Instaliranje dodatnog modula Android Development Tools (ADT)

Dodatni modul ADT instalira se iz razvojnog okruženja Eclipse, ali to morate uraditi prijavljeni kao administrator. Upotrebite prečicu koju ste ranije napravili ili pokrenite *eclipse.exe* direktno iz direktorijuma *eclipse*. U oba slučaja otvorite kontekstni meni (obično pritiskom na desni taster miša) i izaberite opciju „Run as administrator" i prihvatite sva bezbednosna upozorenja. Kada se Eclipse učita, otvorite meni Help i izaberite opciju Install New Software...

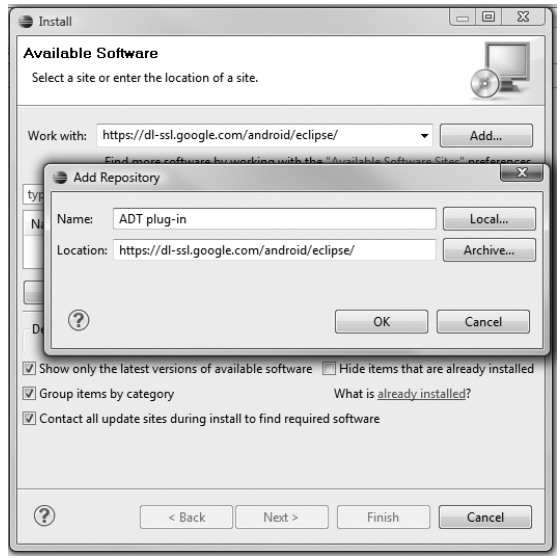
Na ekranu Install unesite sledeću adresu u polje „Work with":

https://dlssl.google.com/android/eclipse

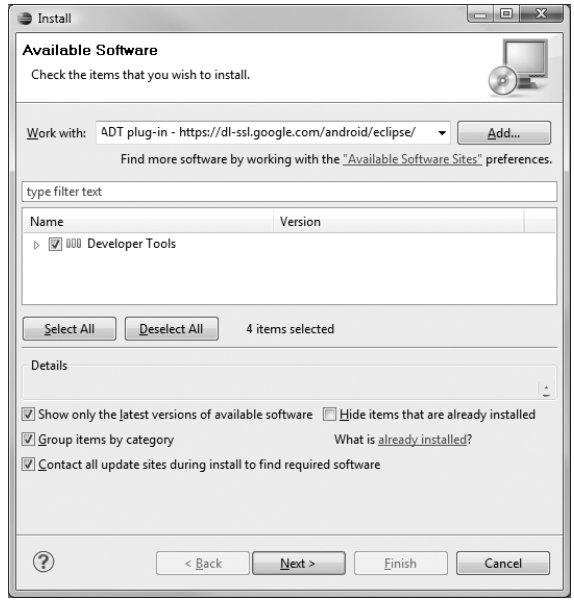
Pritisnite dugme Add. Kada se pojavi ekran Add Repository, u polje Name upišite neko smisleno ime, recimo „ADT plug-in" (navedena adresa će biti prikazana u polju Location ispod njega, videti sliku 1-11).

Pritisnite dugme OK. Ekran će se osvežiti nakon što nakratko prikaže poruku Pending u koloni Name.

Potvrdite polje pored natpisa Developer Tools. Zatim izaberite dugme Next u dnu ekrana (slika 1-12).

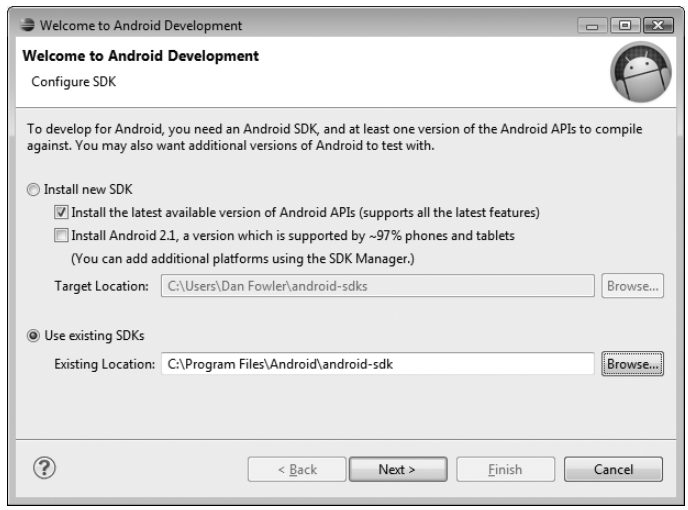


Slika 1-11. Upisivanje lokacije dodatnog modula ADT



Slika 1-12. Biranje šta će se instalirati

Prikazaće se spisak stavki koje će biti instalirane. Ako se pojavi poruka o grešci, proverite da li Eclipse radi s administratorskim nalogom. Pritisnite ponovo Next. Prikazaće se ekran sa uslovima licenci; ispitajte da li su uslovi svih licenci prihvaćeni (izaberite radio-dugme „I accept the terms of the license agreements“). Zatim pritisnite dugme Finish. Da biste završili instalaciju, prihvatite bezbednosno upozorenje; izaberite dugme OK na tom upozorenju (adresa koju ste ranije uneli je zaštićena adresa). Eclipse će zatražiti da ponovo pokrenete računar. Izaberite dugme Restart Now, a Eclipse će se zatvoriti pa ponovo pokrenuti. Pojaviće s dijalog Welcome to Android Development. U polju Existing Location zadajte lokaciju razvojnog kompleta SDK (pošto je SDK Manager već ranije pokretan), pređite u direktorijum za Android SDK (podrazumevani je *C:\Program Files\Android\android-sdk*), pa pritisnite Next (slika 1-13).



Slika 1-13. Povezivanje novoinstaliranog kompleta SDK s novoinstaliranim dodatkom ADT

Pojaviće se pitanje o Googleovom nadgledanju upotrebe razvojnog kompleta Android SDK; izmenite tu opciju ako je potrebno i pritisnite Finish. Razvojno okruženje Eclipse sada je podešeno za prevođenje Android aplikacija i otklanjanje grešaka iz njih. Pogledajte recept 3-3 za konfigurisanje Androidovog emulatora, a zatim i recept 1-4 kao proveru ispravnosti. Priključite fizički uređaj na računar i upotrebite njegove parametre da biste aktivirali mogućnost USB Debugging (ispod Development in Applications).

Videti i

Recept 1-4; recept 1-12; recept 3-3; *http://developer.android.com/sdk/installing.html*, *http://www.eclipse.org/*; *http://www.oracle.com/technetwork/java/javase/downloads/index.html*

1.6 Životni ciklus Android aplikacije

Ian Darwin

Problem

Android aplikacije nemaju metodu „main“; treba da naučite kako se one pokreću i kako se zaustavljaju ili bivaju zaustavljene.

Rešenje

Klasa `android.app.Activity` sadrži nekoliko dobro definisanih metoda životnog ciklusa koje se pozivaju kada se aplikacija pokreće, privremeno zaustavlja, ponovo pokreće itd., kao i metodu koju pozivate da biste aktivnost označili kao završenu.

Objašnjenje

Pošto Android aplikacija pokreće sopstveni Unix proces, uglavnom ne može direktno da utiče na druge aplikacije koje su u toku izvršavanja. Dalvik virtuelna mašina (VM) radi u sprezi sa operativnim sistemom i obaveštava vas kada se aplikacija pokrene, kada korisnik pređe na drugu aplikaciju itd. Postoji dobro definisan životni ciklus za Android aplikacije.

Android aplikacija može biti u jednom od tri moguća stanja:

- Aktivna – aplikacija je vidljiva korisniku i aktivna;
- Pauzirana – aplikacija je delimično zamračena i nema fokus za unošenje podataka;
- Zaustavljena – aplikacija je potpuno izvan vidokruga.

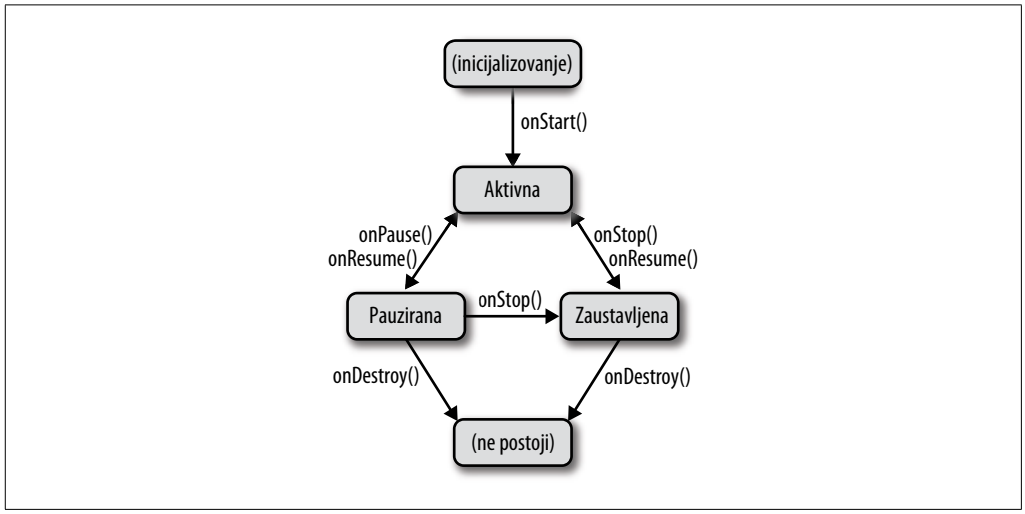
Aplikacije prelaze između navedenih stanja kad god u odgovarajućem trenutku Android pozove za tekuću aktivnost jednu od njenih sledećih metoda:

```
void onCreate(Bundle savedInstanceState)
void onStart()
void onResume()
void onRestart()
void onPause()
void onStop()
void onDestroy()
```

Dijagram stanja za ovaj životni ciklus prikazan je na slici 1-14.

U prvoj aktivnosti aplikacije, metoda `onCreate()` omogućava da saznate da je aplikacija pokrenuta. Uobičajeno je da se u kodu te metode obavljaju konstruktorski poslovi, kao što su izrada „glavnog prozora“ pomoću metode `setContentView()`, pridruživanje osluškivača dugmadima radi pokretanja određenih akcija (uključujući pokretanje novih aktivnosti) itd. Ta metoda je neophodna čak i u najjednostavnijoj Android aplikaciji.

Rezultate izvršavanja pojedinih metoda životnog ciklusa aplikacije možete videti ako u okruženju Eclipse otvorite nov projekat i sve navedene metode redefinišete tako da sadrže naredbe za upisivanje u dnevnik rada aplikacije.



Slika 1-14. Stanja Android aplikacije tokom njenog životnog ciklusa

1.7 Instaliranje .apk datoteka na emulator pomoću alatke ADB

Rachee Singh

Problem

Imate *.apk* datoteku aplikacije i želite da je instalirate na emulator kako biste ispitali rad aplikacije ili zato što to zahteva aplikacija koju razvijate.

Rešenje

Upotrebite alatku ADB sa komandne linije da biste instalirali *.apk* datoteku na tekući aktivan emulator; tu alatku možete iskoristiti i da biste instalirali *.apk* datoteku na povezani fizički Android uređaj.

Objašnjenje

Da biste instalirali *.apk* datoteku, uradite sledeće:

1. Pronađite na svom računaru direktorijum u koji ste instalirali Android SDK. U tom direktorijumu pređite u poddirektorijum *tools*.
2. U poddirektorijumu *tools* potražite izvršnu adb datoteku. Ako takva postoji, to je mesto adb datoteke; u suprotnom, trebalo bi da postoji *.txt* datoteka koja se zove „adb has moved.“ Sadržaj te datoteke samo vas upućuje na mesto gde se nalazi binarna datoteka adb; u *.txt* datoteci stoji da se adb nalazi u direktorijumu *platform-tools* umesto u direktorijumu *tools*.

3. Pošto pronađete program adb, pomoću komande `cd` pređite u njegov direktorijum na terminalu (Linux) ili u prozoru Command Prompt (Windows).
4. Zadajte komandu `adb install` *`lokacija .apk datoteke koju želite da instalirate`*. Ako na Linuxu dobijete poruku „command not found“, pokušajte sa „./adb“ umesto samo „adb“.

Tada bi trebalo da započne instaliranje na tekućem aktivnom uređaju (tj. na emulatoru koji radi na vašem računaru ili na stvarnom Android uređaju koji je povezan na računar).

Kada se završi postupak instaliranja, trebalo bi da u meniju Android uređaja/emulatora vidite ikonicu aplikacije koju ste upravo instalirali (slika 1-15).

1.8 Instaliranje aplikacija na emulator iz skladišta SlideME

David Dawes

Problem

Skladišta aplikacija vrlo su važan razlog privlačnosti savremenih pametnih telefona. Googleov Android Market je zvanično skladište aplikacija, ali postoje i druga koja možete koristiti.

Slika 1-15. Komanda za instaliranje aplikacija

Rešenje

SlideMe LLC (*http://slideme.org/*) jedno je od alternativnih skladišta. Na SlideME je dozvoljeno da instalirate i druge aplikacije (možda želite da integrišete svoju aplikaciju s drugim), kao i da testirate postupak objavljivanja vaših aplikacija i preuzimanje na emuliran Android uređaj. SlideME je dostupan i mnogim korisnicima Androida koji nemaju pristup Google Android Marketu, kao što su ljudi s nepodržanim uređajima i oni koji žive u zemlji koju Android Market ne podržava.

Objašnjenje

Jedna od alternativa zvaničnom Android Marketu je SlideME, alternativno skladište aplikacija (*http://slideme.org/*). SlideME možda nema onoliko aplikacija koliko Googleov Android Market, ali ipak pruža određene prednosti, među kojima je jednostavan rad na emuliranim Android uređajima.

Pređite na veb lokaciju SlideME (*http://slideme.org/*) pomoću svog emuliranog Android uređaja, pretražite ponuđene aplikacije i izaberite neku od besplatnih. Nakon izvesnog vremena potrebnog za preuzimanje datoteke, otvorite je (pomoću male strelice u gornjem levom uglu), proučite uslove licence i pokrenite *.apk* datoteku koju ste upravo preuzeli da biste instalirali aplikaciju. Tokom postupka instalacije, pojaviće se zahtev da pregledate i prihvatite uslove licence za softver.

Pošto instalirate aplikaciju preuzetu sa SlideME, možete pretraživati katalog i instalirati druge aplikacije bez upotrebe čitača veba. To je znatno lakše od preuzimanja aplikacija pomoću čitača veba, pošto je prezentacija projektovana za Android uređaj; samo izaberite odgovarajuću kategoriju i pronađite na listi aplikaciju koju želite da instalirate. Tokom upotrebe aplikacije na mom emulatoru imao sam neke probleme sa stabilnošću – emulator se povremeno „zamrzavao“ – ali uspeo sam da instaliram nekoliko jednostavnih aplikacija, kao što je Grocery List.

Na diskusionom forumu Android Invasion, na Linkedin.com, zapazio sam da su neki korisnici Androida razočarani što mnogi davaoci usluga mobilne telefonije ne uključuju zvanični Android Market u svoju ponudu softvera za Android mobilne telefone i ako niste dovoljno stručni da „rutujete“ i „flešujete“ svoj Android telefon, nema načina da dođete do Android Marketa. Pošto se većina korisnika ne razume u „rutovanje“ i „flešovanje“ mobilnih telefona, SlideME im pruža alternativni način da dođu do besplatnih i jeftinih aplikacija za svoje telefone.

Videti i

SlideME vam omogućava i da objavite svoju aplikaciju u tom skladištu aplikacija; pređite na stranicu Applications veb lokacije SlideME (*http://slideme.org/applications*).

Više informacija o razvijanju aplikacija za SlideME naći ćete na adresi *http://slideme.org/developers*.

1.9 Deljenje Java klasa s drugim projektom u okruženju Eclipse

Ian Darwin

Problem

Želite da koristite klasu iz drugog projekta, ali ne želite da je odatle samo kopirate i umetnete u nov projekat.

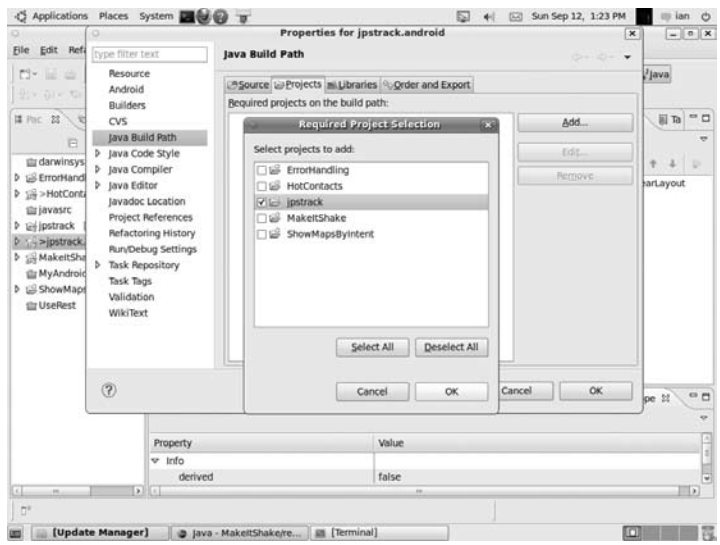
Rešenje

Projekat kojem klasa pripada dodajte kao „referenciran projekat“, a Eclipse (i DEX) će uraditi šta treba.

Objašnjenje

Često se događa da se klasa iz jednog projekta koristi i u više drugih. U mom programu za GPS praćenje, JPSTrack, Android verzija „pozajmljuje“ neke klase od verzije Java SE, kao što je modul za ulazno/izlazne operacije sa datotekama. Sigurno ne biste želeli da proizvoljno kopirate i umećete klase iz jednog projekta u drugi jer bi onda održavanje postalo nemoguće.

U najjednostavnijem slučaju, kada bibliotečki projekat sadrži izvorni kôd klasa koje želite da uvezete, samo treba da projekat koji sadrži potrebne klase (verzija Java SE u ovom slučaju) dodate kao referenciran u putanju za sklapanje projekta. Izaberite Project → Properties → Java Build Path, izaberite opciju Projects i pritisnite dugme Add. Na slici 1-16, dodajem SE projekat „jpstrack“ kao komponentu od koje zavisi projekat „jpstrack.android“.

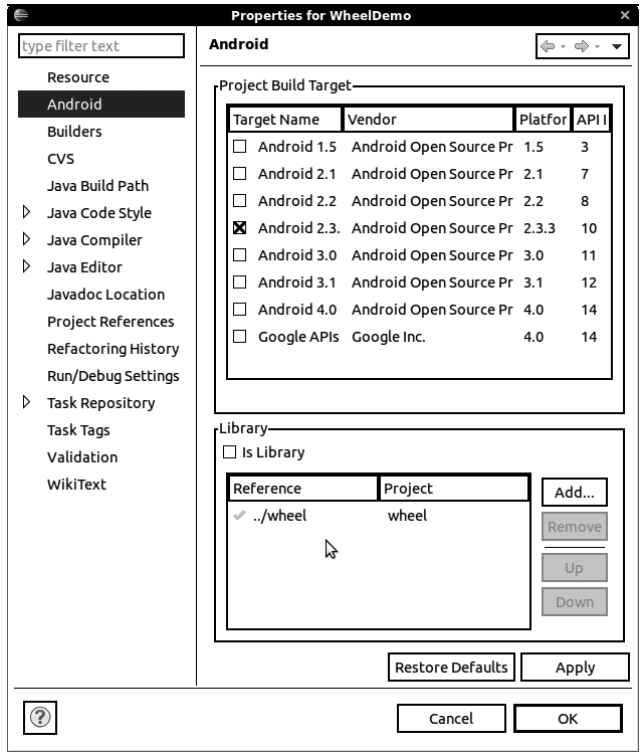


Slika 1-16. Povezivanje jednog projekta kao dela od kojeg zavisi drugi projekat – u standardnom razvojnom okruženju Eclipse

Trebalo bi da autori aplikacija za mobilne uređaje i na drugim ciljnim platformama imaju u vidu da opisana tehnika nije primenjiva ako na svom primerku okruženja Eclipse imate instaliran i tekući (kraj 2011. godine) dodatni modul BlackBerry Java. Razlog je greška u modulu BlackBerry Java; čak i projekte koji nisu za BlackBerry uređaje pogrešno obeležava kao da zavise od ne-BlackBerry bibliotečkih projekata, a projekat označava kao da sadrži grešku, što sprečava ispravno generisanje i izvršavanje koda. Uklonite modul s greškom, ili ga instalirajte u vlastiti primerak okruženja Eclipse.

Druga mogućnost je da JAR datoteku napravite pomoću alatke Ant ili čarobnjaka iz okruženja Eclipse. Drugi projekat treba da referencira tu JAR datoteku kao spoljnu JAR datoteku koju zadate parametrom `classpath`. Ili, fizički kopirajte JAR datoteku u direktorijum *libs* i referencirajte je odatle.

Novija metoda, koja je često pouzdanija i sada se zvanično preporučuje, ali je primenjiva samo ako su oba projekta Android projekti, jeste da prvu biblioteku deklarišete kao bibliotečki projekat, u odeljku Project → Properties → Android → Library, i da zatim pomoću dugmeta Add u drugom projektu na istom ekranu zadate bibliotečki projekat kao zavisan od glavnog projekta (slika 1-17).



Slika 1-17. Zadavanje jednog projekta kao zavisnog od drugog – pomoću alatke ADT

Za ljubitelje komandne linije, prva metoda podrazumeva menjanje sadržaja datoteke *.classpath*, dok druga metoda samo umeće nove stavke u datoteku *project.properties*, na primer:

```
# Project target
target=android-7
android.library=false
android.library.reference.1=../wheel
```

Pošto verovatno oba projekta držite pod kontrolom nekog programa za upravljanje izvornim kodom (a to bi svakako trebalo da činite ako nameravate da te programe jednog dana objavite na tržištu!), ne zaboravite da „obeležite“ oba projekta kada objavite Android projekat – jedna od prednosti zbog kojih treba koristiti programe za upravljanje izvornim kodom jeste mogućnost da rekonstruišete tačno ono što ste objavili.

Videti i

Zvanična dokumentacija o bibliotečkim projektima – Library Projects (*http://developer.android.com/guide/developing/projects/index.html#LibraryProjects*).

1.10 Referenciranje biblioteka radi implementiranja spoljne funkcionalnosti

Rachee Singh

Problem

Potrebno vam je da u izvornom kodu referencirate spoljnu biblioteku.

Rešenje

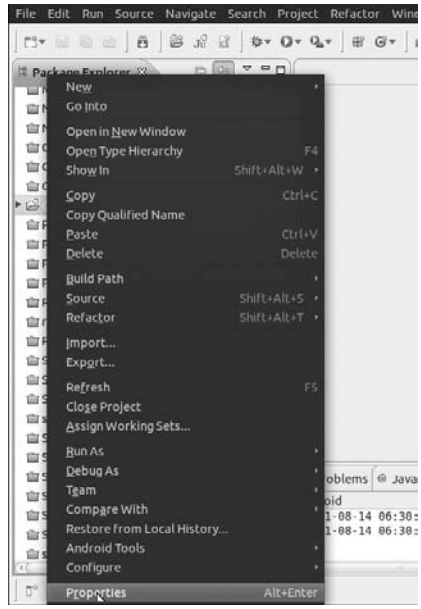
Nabavite JAR datoteku biblioteke koja vam treba i dodajte je u svoj projekat.

Objašnjenje

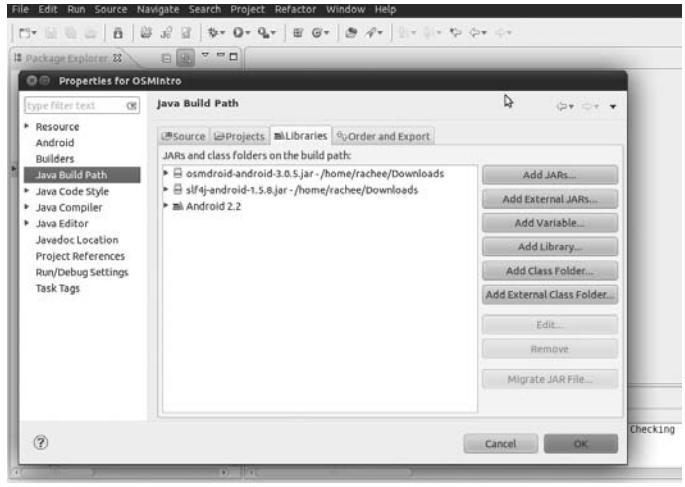
Primera radi, za vašu aplikaciju vam treba AndroidPlot, biblioteka za crtanje dijagrama i grafikona, ili OpenStreetMap, wiki projekat koji generiše i stavlja na raspolaganje besplatne geografske podatke i mape. U tom slučaju, vaša aplikacija treba da referencira te biblioteke. To možete obaviti u okruženju Eclipse, u nekoliko jednostavnih koraka:

1. Preuzmite JAR datoteku biblioteke koju želite da koristite.
2. Pošto napravite Android projekat aplikacije u okruženju Eclipse, desnim tasterom miša pritisnite ime projekta i u meniju izaberite Properties (slika 1-18).
3. Na listi na levoj strani izaberite opciju Java Build Path a zatim mišem pritisnite jezičak Libraries.
4. Pritisnite dugme Add External JARs.
5. Zadajte direktorijum u kojem se nalazi JAR datoteka biblioteke.

U ovoj fazi projektu se dodaje direktorijum *Referenced Libraries* u kojem se nalaze JAR datoteke koje ste dodali (slika 1-19).



Slika 1-18. Biranje svojstava projekta

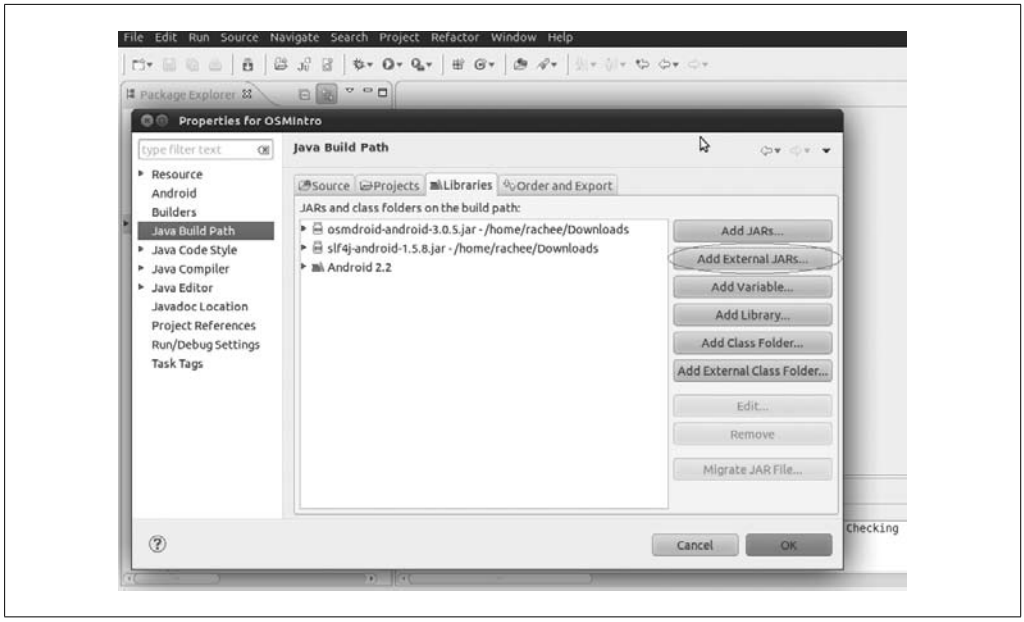


Slika 1-19. Dodavanje biblioteke projektu

Druga mogućnost je da u glavnom projektu napravite direktorijum *lib*, fizički kopirate JAR datoteke u njega i da ih zatim pojedinačno dodate u projekat jednu za drugom kao u prethodnom slučaju, ali bez pritiskanja dugmeta Add JARs. Tako ćete sve imati na jednom mestu (što je naročito dobro ako se projekat preko sistema za upravljanje verzijama deli s drugim projektima koji rade pod drugim operativnim sistemom i nisu u stanju da pronađu JAR datoteke na istom mestu). Međutim, time se povećava odgovornost za licenciranje JAR datoteka uključenih u projekat. Videti sliku 1-20.

U oba slučaja, ako projekat sklapate pomoću alatke Ant, obavezno ažurirajte datoteku *build.xml*.

Koju god opciju da izaberete, dodavanje biblioteka projektu vrlo je jednostavno.



Slika 1-20. Dodavanje spoljne JAR datoteke

1.11 Upotreba primera iz kompleta Android SDK radi izbegavanja pronalaženja „rupe na saksiji“

Daniel Fowler

Problem

Ponekad treba uložiti mnogo napora da bi se kodirala određena funkcionalnost, naročito kada je dokumentacija štura ili ne sadrži nikakve primere.

Rešenje

Proučavanje postojećeg koda biće od koristi. Android SDK sadrži primere programa koje možete ispitati da biste videli kako rade.

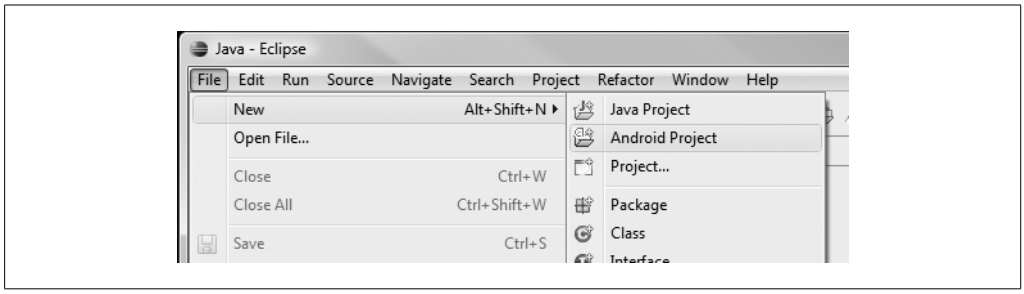
Objašnjenje

Uz Android SDK isporučuje se više primera aplikacija koji vam mogu koristiti kada pokušavate da kodirate određenu funkcionalnost. Proučavanje primera koda može biti vrlo poučno. Pošto instalirate Android SDK, na raspolaganju imate više primera:

- Accelerometer Play
- Accessibility Service
- API Demos
- Backup and Restore
- Bluetooth Chat
- Business Card
- Contact Manager
- Cube Live Wallpaper
- Home
- Honeycomb Gallery
- JetBoy
- Lunar Lander
- Multiple Resolutions
- Near Field Communication
- Note Pad
- RenderScript
- Sample Sync Adapter
- Searchable Dictionary
- Session Initiation Protocol
- Snake
- Soft Keyboard
- Spinner
- SpinnerTest
- StackView Widget
- TicTacToeLib
- TicTacToeMain

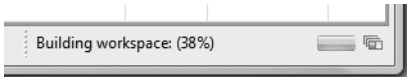
- USB
- Wiktionary
- Wiktionary (Simplified)
- Weather List Widget
- XML Adapters

Da biste iz okruženja Eclipse otvorili primer projekta, otvorite meni File a zatim izaberite opciju Android Project (slika 1-21).



Slika 1-21. Otvaranje novog Android projekta

U dijalogu New Android Project, izaberite opciju „Create project from existing sample". Pritisnite Next pa izaberite ciljni API nivo (Build Target). Prikazaće se lista raspoloživih primera za izabrani ciljni API nivo. Ako traženi primer nije na listi, vratite se i izaberite drugi ciljni API nivo (Build Target). (Primer možda nije instaliran; dodatni primeri se mogu instalirati pomoću SDK Managera ako su bili izostavljeni pri instaliranju kompleta Android SDK.) Izaberite primer pa pritisnite Finish da bi se primer kopirao u radni prostor i sklopio (na statusnoj traci se prikazuje napredovanje postupka).



Nakon kraćeg vremena, primer je spreman za izvršavanje a vi ćete moći da pretražujete izvorni kôd da biste videli kako radi.

Ako su primeri aplikacija uklonjeni iz SDK direktorijuma *samples*, otvorite traženi primer pomoću opcije „Create project from existing source" dijaloga New Android Project.

Kada primer aplikacije pokrećete po prvi put, izaberite opciju Android Application u dijalogu Run As koji će se možda otvoriti. Osim toga, može biti potrebno da konfigurišete odgovarajući AVD kako biste mogli da izvršite primer (videti recept 3.3). Videti sliku 1-22.

Slika 1-22. Aplikacija API u akciji

Videti i

Veb lokaciju Android Developers na adresi *http://developer.android.com/index.html*; ovu knjigu, naravno.

Na vebu možete pronaći i dodatne programe ili primere. Ako ne možete da nađete ono što vam treba, potražite pomoć na lokaciji Stack Overflow (*http://www.stackoverflow.com*; zadajte „android“ kao pojam) ili na IRC (Internet Relay Chat) kanalu #androiddev servera freenode.

1.12 Održavanje razvojnog kompleta Android SDK u ažurnom stanju

Daniel Fowler

Problem

SDK se mora održavati u ažurnom stanju kako bi programeri mogli da rade s najnovijim API-jima Android platforme koja se neprestano menja.

Rešenje

Upotrebite alatku Android SDK Manager da biste ažurirali postojeće instalirane SDK pakete i instalirali nove SDK pakete. To važi i za paket iz trećih izvora koji obezbeđuju funkcionalnost specifičnu za određene ciljne uređaje.

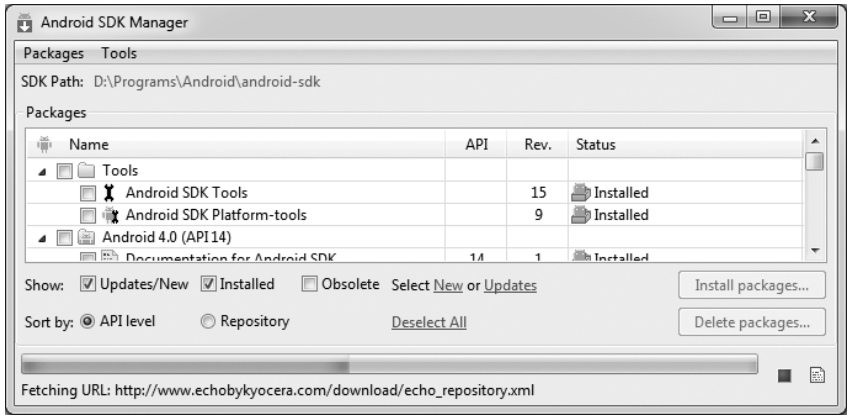
Objašnjenje

Pošto operativni sistem (OS) Android neprestano evoluira, to se događa i sa razvojnim kompletom Android SDK. Razvoj i promene Androida podstiču sledeći elementi:

- Googleova istraživanja i razvoj
- Proizvođači telefona koji razvijaju nove i poboljšane uređaje
- Rešavanje bezbednosnih problema i sprečavanje da neko iskoristi moguće bezbednosne propuste
- Potreba da se podrže nove vrste uređaja (na primer, podrška za tablične uređaje koja je dodata u verziji 3.0)
- Podrška za nove hardverske interfejse (na primer, u verziji 2.3 dodata je podrška za NFC tehnologije).
- Ispravljanje otkrivenih grešaka
- Poboljšanja funkcionalnosti (na primer, novi JavaScript motor)
- Izmene u Linuxovom jezgru
- Zastarevanje interfejsa za programiranje koji postaju suvišni
- Novi načini upotrebe (na primer, Google TV)
- Šira zajednica autora koji razvijaju aplikacije za Android

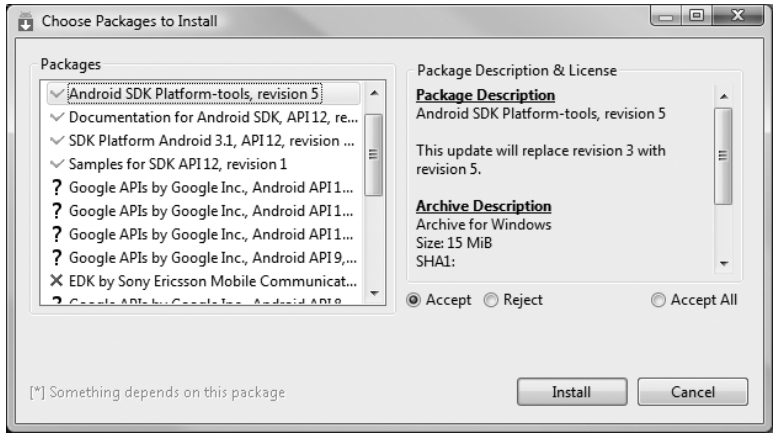
Instaliranje razvojnog kompleta Android SDK opisano je na drugom mestu (videti recept 1.5 ili na *http://developer.android.com/sdk/installing.html*). Pošto programer instalira SDK na razvojni računar i okruženje za programiranje počne da radi kako treba, s vremena na vreme treba da proveri treba li ažurirati SDK.

SDK možete održavati u ažurnom stanju pomoću programa SDK Manager. (Na Windows računaru pokrenite datoteku *SDK Manager.exe* iz direktorijuma *C:\Program Files\Android\androidsdk*, ili pritisnite dugme Start, a zatim izaberite All Programs → Android SDK Tools, pa pritisnite stavku SDK Manager). Možete ga pokrenuti i iz razvojnog okruženja Eclipse (u meniju Window izaberite stavku Android SDK Manager). Videti sliku 1-23. Razvojni komplet Android SDK podeljen je u više paketa. SDK Manager automatski ispituje da li postoje ispravke za postojeće pakete i prikazuje listu novih paketa, kao i paketa koji potiču od proizvođača konkretnih uređaja.



Slika 1-23. Program Android SDK Manager

Ispravke (kao i opcioni paketi na raspolaganju) prikazuju se u obliku liste. Ako za određenu ispravku ili paket postoje uslovi licence koje treba prihvatiti, prikazuje se sa znakom pitanja. Istaknite svaki paket koji je označen znakom pitanja da biste pročitali uslove licence. Paket možete prihvatiti ili odbaciti pomoću radio-dugmadi. Odbačeni paketi se obeležavaju crvenim znakom × (slika 1-24).

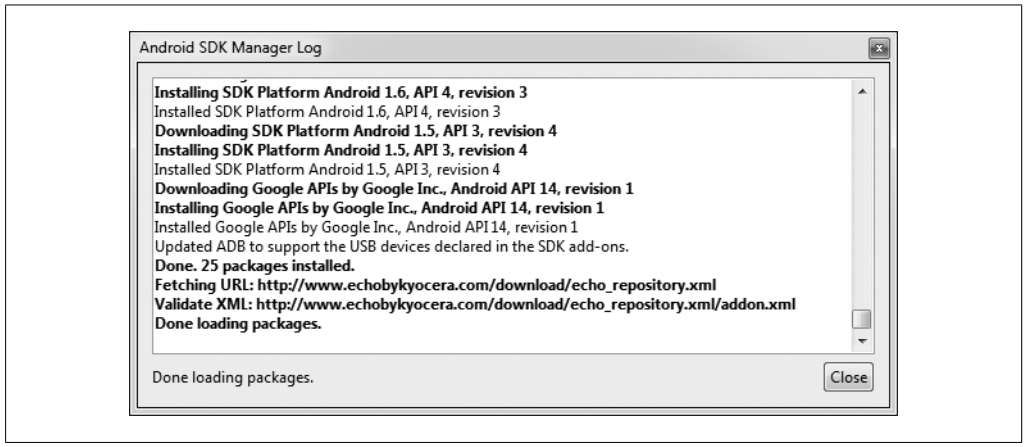


Slika 1-24. Biranje SDK paketa

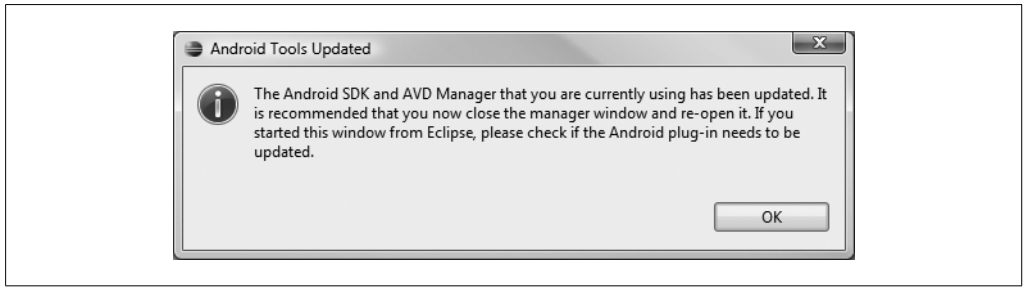
Druga mogućnost je da izaberete opciju Accept All kako biste prihvatili sve uslove za sve pakete i ispravke. Svi paketi i ispravke koji su spremni za preuzimanje i instaliranje prikazuju se obeleženi zelenim znakom potvrde.

Pritisnite dugme Install da biste započeli postupak preuzimanja i instaliranja; kada se postupak završi, pritisnite dugme Close (slika 1-25).

Ako je i sam program SDK Manager ažuriran, pojaviće se poruka da treba ponovo da ga pokrenete (slika 1-26).



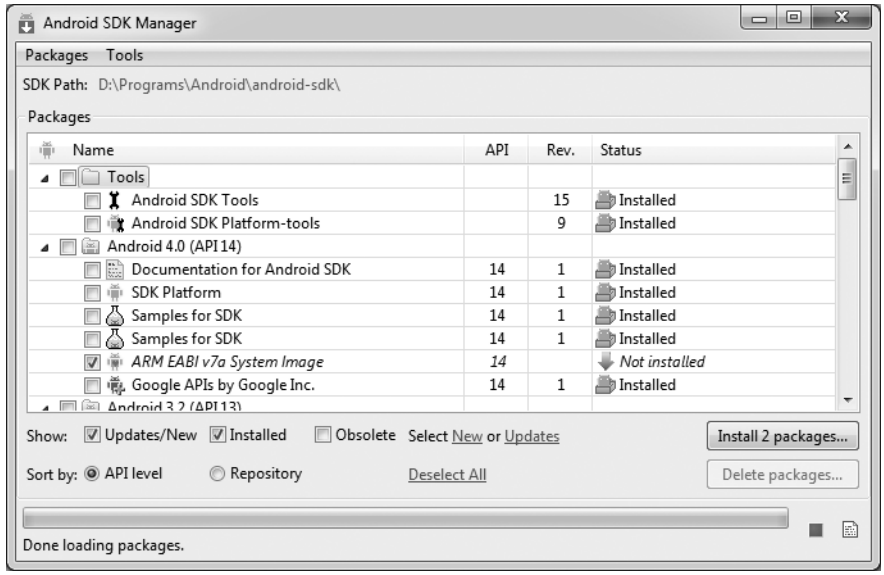
Slika 1-25. Prozor SDK Manager Log



Slika 1-26. Poruka o ažuriranju SDK Managera

SDK Manager služi i za preuzimanje dodatnih paketa koji nisu deo standardne platforme. Taj mehanizam koriste proizvođači pojedinih uređaja da bi pružili podršku za svoj hardver. Na primer, LG Electronics nudi 3D uređaj, za koji postoji dodatan paket pomoću kojeg aplikacije dobijaju podršku za 3D mogućnosti. Taj mehanizam koristi i Google da bi omogućio preuzimanje opcionih API-ja.

U dijalogu SDK Manager, na listi na levoj strani stavite znakove potvrde ispred paketa koji vam trebaju a zatim pritisnite dugme Install (slika 1-27). Ako se određeni paket iz trećeg izvora ne nalazi na listi, potrebno je da pomoću menija Tools upišete URL paketa, koji obezbeđuje izdavač paketa, u datoteku *respository.xml*.



Slika 1-27. Lista instaliranih komponenata i onih koje se mogu instalirati

Moguće greške pri ažuriranju na Windowsu

U jednom tako složenom sistemu, mnogo toga može krenuti naopako. U ovom odeljku razmatramo neke moguće probleme i rešenja za njih.

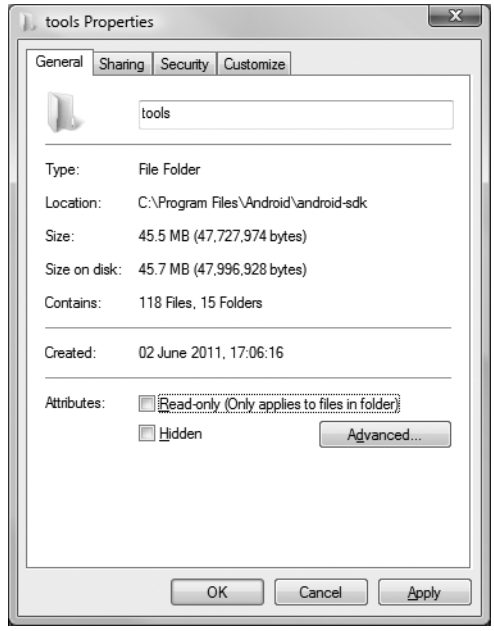
Pokrenite SDK Manager sa ovlašćenjima administratora Na Windows računaru, podrazumevani direktorijum za SDK je *C:\Program Files\Android\android-sdk*. To je direktorijum kojem je ograničen pristup što može biti uzrok neuspeha pri instaliranju SDK. Može se pojaviti dijalog s porukom čiji je naslov „SDK Manager: failed to install“ (slika 1-28).

Slika 1-28. Poruka SDK Manager: Failed to install

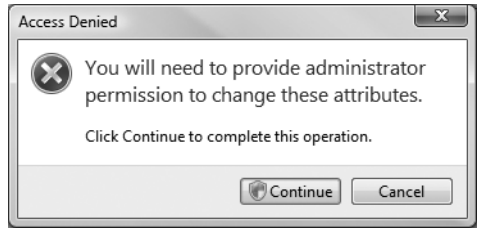
Da biste izbegli ovu grešku, proverite sledeće:

- Izvucite utikače Android uređaja (to može sprečiti zatvaranje programa *adb.exe*).
- Pređite u direktorijum *C:\Program Files\Android\Android-sdk* i otvorite prozor Properties za direktorijum *tools* (desnim tasterom miša otvorite kontekstni meni, a zatim izaberite stavku Properties). Uklonite znak potvrde iz polja „Read-only (Only applies to files in folder)“, ako ga ima (slika 1-29).

Može biti potrebno da dozvolite menjanje vrednosti tih atributa (slika 1-30).



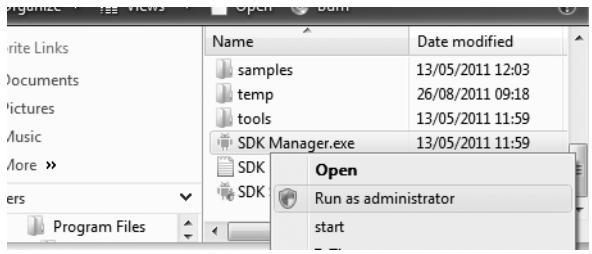
Slika 1-29. Podešavanje atributa za čitanje/pisanje pod Windowsom



Slika 1-30. Potvrđivanje ovlašćenja

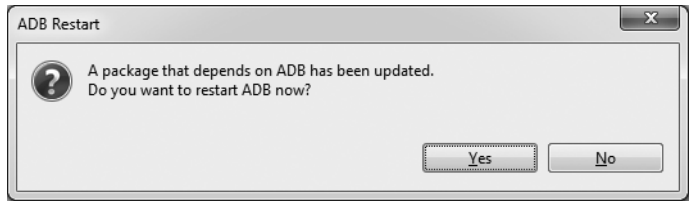
Pojaviće se dijalog Confirm Attribute Changes; potvrdite opciju „Apply changes to this folder, subfolders and files“ i pritisnite dugme OK. Zatim uradite sledeće:

- Ponovo pokrenite računar.
- Proverite da li su svi ostali programi zatvoreni, naročito File Explorer.
- Pokrenite *SDK Manager.exe* pod administratorskim nalogom. Otvorite kontekstni meni i izaberite opciju „Run as administrator“ (slika 1-31).



Slika 1-31. Opcija Run as administrator

Zatvorite ADB pre ažuriranja Može se pojaviti poruka sa zahtevom da ponovo pokrenete alatku ADB (Android Debugger) (slika 1-32).

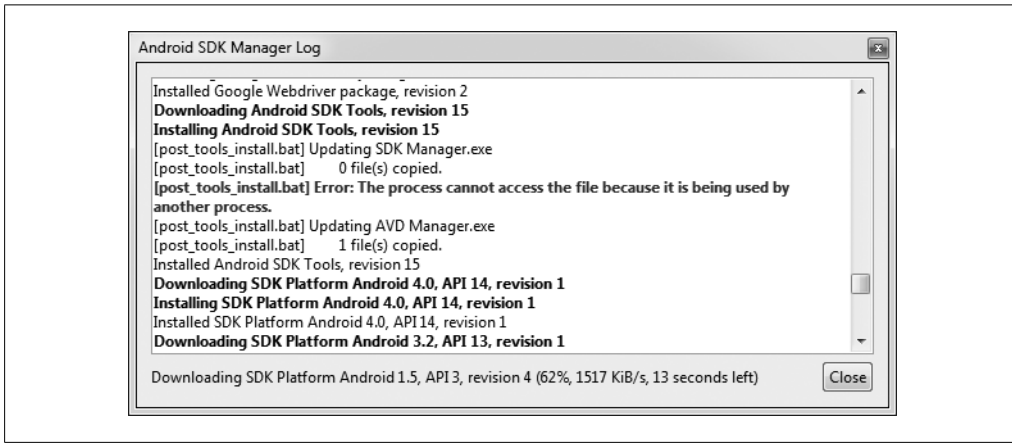


Slika 1-32. Potvrđivanje za ponovno pokretanje alatke ADB

Najbolje je da dok radi SDK Manager, program ADB ne bude aktivan, a ne bi ni trebalo da bude pokrenut ako je Windows upravo podignut. Druga mogućnost je da zaustavite adb.axe pomoću Windowsove alatke Task Manager. Na ovu poruku odgovorite sa No ako ADB nije aktivan; u suprotnom, odgovorite sa Yes.

SDK Manager ne uspeva da sam sebe ažurira Tokom instaliranja ispravki za program SDK Manager može doći do greške (slika 1-33).

Da biste je otklonili, proverite da li su svi drugi programi zatvoreni (uključujući i *adb.exe*). Zatim kopirajte datoteku *SDK Manager.exe* iz direktorijuma *C:\Program Files\Android\android-sdk\tools\lib* u *C:\Program Files\Android\android-sdk* (ili tamo gde je SDK instaliran). Zatim ponovo pokrenite SDK Manager (slika 1-32).



Slika 1-33 Prozor Android SDK Manager Log

Ažuriranje razvojnog okruženja Eclipse Pošto ažurirate Android SDK i otvorite okruženje Eclipse, može se pojaviti sledeće upozorenje (slika 1-34).

Slika 1-34. Android SDK version incorrect

U okruženju Eclipse, izaberite meni Help a zatim stavku Check for Updates. Sačekajte da se završi postupak pregledanja ispravki okruženja Eclipse za Android i one se prikažu. Pritisnite dugme Next dvaput jedno za drugim i prihvatite uslove licence. Zatim pritisnite dugme Finish da biste započeli postupak preuzimanja i ažuriranja. Može se pojaviti upozorenje o nepotpisanom sadržaju. Pritisnite dugme OK da biste prihvatili upozorenje (samo ako ažurirate iz okruženja Eclipse). Kada se instaliranje ispravki završi, ponovo pokrenite okruženje Eclipse (pojaviće se poruka da to uradite).

Dodatne informacije o otklanjanju grešaka pri ažuriranju SDK Managera i Android dodatni modul za Eclipse naći ćete na veb lokaciji Android Developers.

Videti i

Recept 1.5; Instaliranje kompleta Android SDK (*http://developer.android.com/sdk/installing.html*); Dodavanje komponenata u SDK (*http://developer.android.com/sdk/adding-components.html*); ADT dodatni modul za Eclipse (*http://developer.android.com/sdk/eclipse-adt.html*)

1.13 Izrada slike sadržaja ekrana emulatora / Android uređaja

Rachee Singh

Problem

Želite da napravite sliku sadržaja ekrana aplikacije koja radi na Android uređaju.

Rešenje

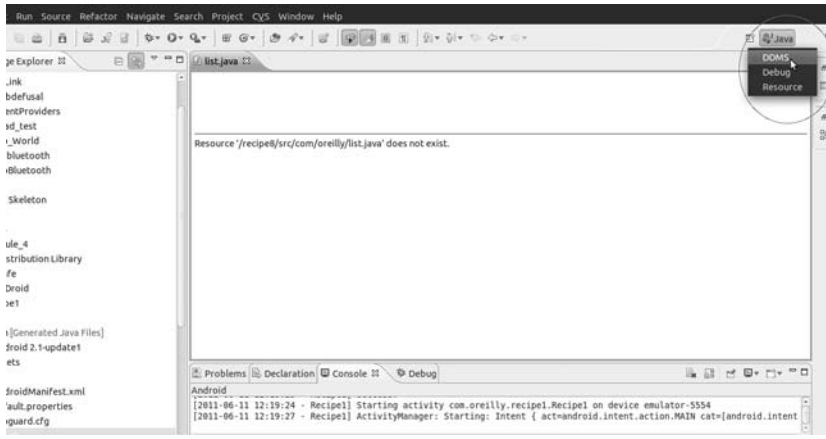
Upotrebite opciju Device Screen Capture u prikazu DDMS (Dalvik Debug Monitor Server), u okruženju Eclipse.

Objašnjenje

Da biste iskoristili opciju Device Screen Capture, uradite sledeće:

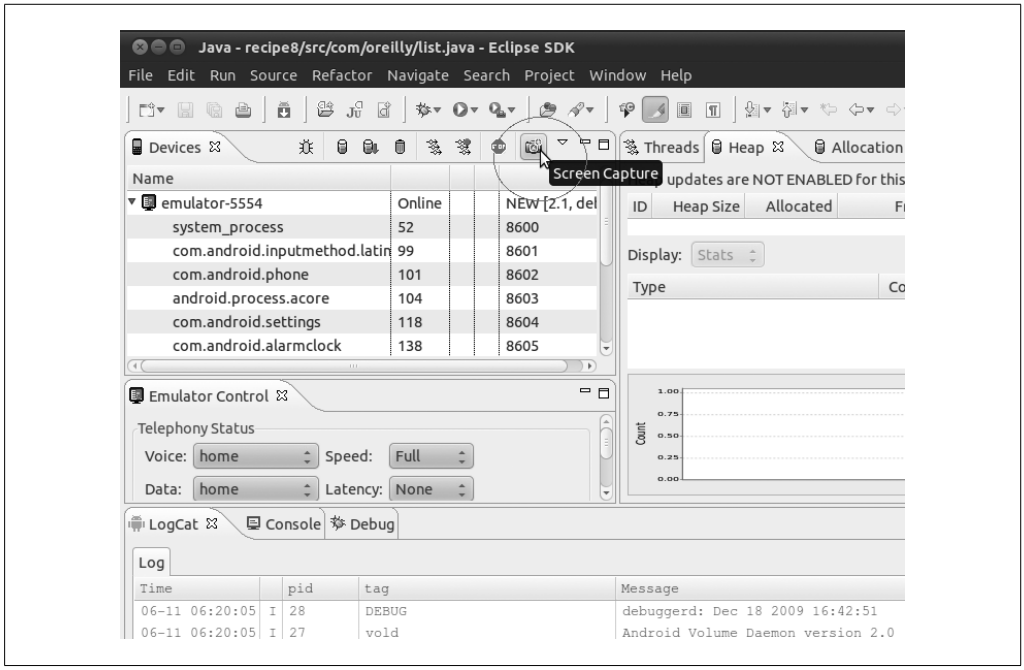
1. Pokrenite aplikaciju u okruženju Eclipse i pređite u prikaz DDMS (meni Window →Open Perspective → Other → DDMS) ili meni Window → Show → Other → Android → Devices; prethodni način je prikazan na slici 1-36).

   Na slici 1-35 natpis „Resource...does not exist“ (resurs... ne postoji) prikazuje se samo zato što je prethodno u okruženju Eclipse bio zatvoren projekat, što ni u čemu ne utiče na korake postupka koji ovde opisujemo.

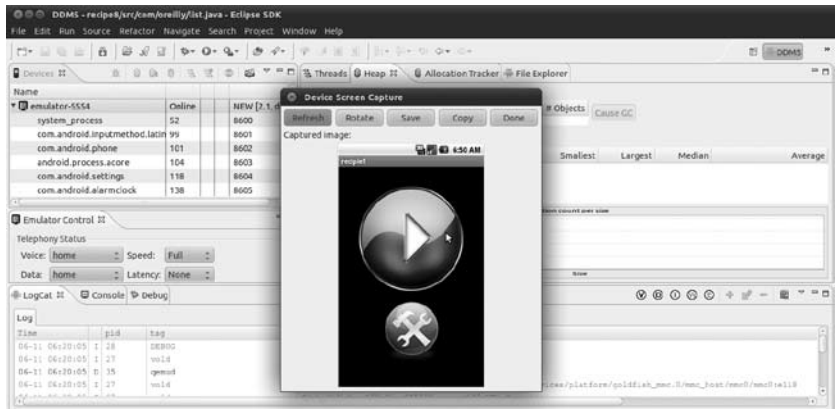


Slika 1-35. Otvaranje DDMS prikaza

2. U prikazu DDMS, izaberite emulator uređaja za koji vam je potrebna slika ekrana.
3. U prikazu DDMS, mišem pritisnite ikonicu Screen Capture (slika 1-36).
4. Otvoriće se prozor u kojem je prikazan tekući izgled ekrana emulatora/Android uređaja. Trebalo bi da liči na onaj prikazan na slici 1-37. Tu sliku ekrana možete snimiti na disk i iskoristiti je za opisivanje aplikacije!



Slika 1-36. Izrada slike ekrana na uređaju



Slika 1-37. Slika ekrana

Videti i

U nekim distribucijama postoje i drugi načini izrade slika ekrana. CyanogenMod 7.x (*http://cyanogenmod.com/*) to omogućava u meniju koji dobijete kada dugačko pritisnete dugme za uključivanje napajanja. Neki HTC-ovi tablični računari opremljeni pisaljkama omogućavaju izradu slike ekrana iz menija Pen. Ice Cream Sandwich (Android 4.0) ima ugrađen mehanizam za preuzimanje slike ekrana na fizičkim uređajima: samo pritisnite kontrolu za utišavanje zvuka istovremeno s dugmetom za uključivanje napajanja; slika se smešta na uređaj i možete da je vidite pomoću aplikacije Gallery.

1.14 Program: jednostavan primer tajmera koji odbrojava

Wagied Davids

Problem

Treba vam jednostavan tajmer koji odbrojava, tj. program koji će odbrojavati zadati broj sekundi dok ne dođe do nule.

Rešenje

U Androidu postoji ugrađena klasa `CountDownTimer` koja omogućava izradu tajmera sa odbrojavanjem. Jednostavno se koristi, efikasna ja i radi (što se podrazumeva!).

Objašnjenje

Postupak izrade tajmera koji odbrojava sastoji se od sledećih koraka:

1. Napravite klasu izvedenu od klase `CountDownTimer`. Konstruktor te klase prihvata dva argumenta, `CountDownTimer(long millisInFuture, long countDownInterval)`. Prvi je broj milisekundi od sada pa do isteka intervala odbrojavanja; u tom trenutku se poziva metoda `onFinish()` izvedene klase. Drugi argument pokazuje svakih koliko milisekundi želite da dobijate obaveštenja da je tajmer i dalje aktivan, najčešće zato da biste ažurirali neki pokazivač napredovanja ili komunicirali s korisnikom u nekom drugom obliku.

   Metoda `onTick()` izvedene klase poziva se kad god istekne broj milisekundi zadat ovim parametrom.
2. Redefinišite metode `onTick()` i `onFinish()`.
3. Napravite novu instancu vaše Android aktivnosti.
4. Pozovite metodu `start()` te instance.

Primer programa za tajmer s odbrojavanjem sastoji se od XML datoteke sadržaja ekrana (primer 1-4) i odgovarajućeg Java koda (primer 1-5). Kada ga pokrenete, trebalo bi da bude nalik onome na slici 1-38, mada će se vremena verovatno razlikovati.

Primer 1-4. Datoteka main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <Button
        android:id="@+id/button"
        android:text="Start"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" />
    <TableLayout
        android:padding="10dip"
        android:layout_gravity="center"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content">
        <TableRow>
            <TextView
                android:id="@+id/timer"
                android:text="Time: "
                android:paddingRight="10dip"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content" />
            <TextView
                android:id="@+id/timeElapsed"
                android:text="Time elapsed: "
                android:paddingRight="10dip"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content" />
        </TableRow>
    </TableLayout>
</LinearLayout>
```

Primer 1-5. Datoteka Main.java

```
package com.examples;

import android.app.Activity;
import android.os.Bundle;
import android.os.CountDownTimer;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;

public class Main extends Activity implements OnClickListener
    {
        private MalibuCountDownTimer countDownTimer;
        private long timeElapsed;
        private boolean timerHasStarted = false;
        private Button startB;
        private TextView text;
        private TextView timeElapsedView;
```

```
private final long startTime = 50 * 1000;
private final long interval = 1 * 1000;

/** Poziva se pri inicijalizovanju aktivnosti. */
@Override
public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        startB = (Button) this.findViewById(R.id.button);
        startB.setOnClickListener(this);

        text = (TextView) this.findViewById(R.id.timer);
        timeElapsedView = (TextView) this.findViewById(R.id.timeElapsed);
        countDownTimer = new MalibuCountDownTimer(startTime, interval);
        text.setText(text.getText() + String.valueOf(startTime));
    }

@Override
public void onClick(View v)
    {
        if (!timerHasStarted)
            {
                countDownTimer.start();
                timerHasStarted = true;
                startB.setText("Start");
            }
        else
            {

                countDownTimer.cancel();
                timerHasStarted = false;
                startB.setText("RESET");
            }
    }

// klasa izvedena od klase CountDownTimer
public class MalibuCountDownTimer extends CountDownTimer
    {

        public MalibuCountDownTimer(long startTime, long interval)
            {
                super(startTime, interval);
            }

        @Override
        public void onFinish()
            {

                text.setText("Time's up!");
                timeElapsedView.setText("Time Elapsed: „ +
                    String.valueOf(startTime));
            }
```

```
            @Override
            public void onTick(long millisUntilFinished)
                {
                    text.setText("Time remain:" + millisUntilFinished);
                    timeElapsed = startTime - millisUntilFinished;
                    timeElapsedView.setText("Time Elapsed: " +
                        String.valueOf(timeElapsed));
                }
        }
}
```

Slika 1-38. Resetovanje tajmera

URL za preuzimanje izvornog koda

Izvorni kôd za ovaj primer možete preuzeti na veb lokaciji izvornog izdanja ove knjige, na *http://github.com/AndroidCook/Android-Cookbook-Examples*, iz poddirektorijuma CountDownTimerExample (videti odeljak „Preuzimanje i upotreba primera koda“ na stranici xvii).

1.15 Program: Tipster, kalkulator iznosa napojnice za OS Android

Sunit Katkar

Problem

Kada izađete s prijateljima u restoran i želite da podelite iznos računa i napojnice, možete zapasti u mnogo ručnih proračuna i neslaganja. Da biste to izbegli, potrebna vam je aplikacija koja omogućava da ukupnom iznosu računa dodate iznos napojnice i da rezultat toga podeli s brojem prisutnih na večeri. Tipster je implementacija te ideje na Androidu, a ovde ga navodimo kao primer kompletne aplikacije.

Rešenje

Ovo je jednostavna vežba u kojoj koristimo osnovne elemente Androidovog grafičkog korisničkog interfejsa (GUI) i povezujemo ih u jednu celinu pomoću jednostavnih proračuna i koda koji obrađuje događaje u komponentama interfejsa. Koristićemo sledeće GUI komponente:

`TableLayout`

Pruža dobru kontrolu nad sadržajem ekrana. Taj raspored omogućava razmeštanje komponenata po istom modelu kao HTML oznaka `Table`.

`TableRow`

Definiše red u prikazu `TableLayout`. Deluje na isti način kao kombinacija HTML oznaka `TR` i `TD`.

`TextView`

Generiše natpis koji omogućava prikazivanje statičkog teksta na ekranu.

`EditText`

Generiše polje za tekst u koje se mogu upisivati vrednosti.

`RadioGroup`

Formira grupu radio-dugmadi.

`RadioButton`

Generiše radio-dugme.

`Button`

Generiše standardno dugme za pritiskanje.

`View`

Objekat tipa `View` iskoristićemo za izradu vizuelnog separatora pomoću određenih atributa za visinu i boju.

Objašnjenje

U Androidu se raspored komponenata na ekranu definiše pomoću XML datoteka. U našem projektu, dodatni modul ADT za okruženje Eclipse generiše datoteku *main.xml* za komponente na ekranu. Ta datoteka sadrži XML definicije pojedinih komponenata i kontejnera u kojima se one nalaze.

Postoji i datoteka *strings.xml* koja sadrži definicije svih resursa znakovnog tipa koji se koriste u aplikaciji. Ikonica aplikacije se nalazi u podrazumevanoj datoteci *icon.png*.

Dalje, imamo datoteku *R.java*, koja se automatski generiše (a tako i ažurira kad god načinite izmenu u datoteci *main.xml*). Ta datoteka sadrži konstante definisane za svaki raspored komponenata i komponentu. Nemojte ručno menjati tu datoteku; to radi modul ADT za vas kad god napravite neku izmenu u XML datotekama projekta.

U našem primeru, datoteka *Tipster.java* je glavna datoteka Java koda aktivnosti.

Recept 1.4 kao i razni Googleovi kursevi opisuju kako se koristi dodatni modul za Eclipse. Pomoću tog modula otvorite u okruženju Eclipse nov Android projekat koji ćete praviti i nazovite ga Tipster. Konačan rezultat biće projekat čija bi struktura trebalo da bude nalik onoj na slici 1-39 (strana 60).

Raspored elemenata na ekranu i postavljanje komponenata

Konačan cilj je izrada ekrana nalik onom na slici 1-39. Za izradu tog ekrana upotrebićemo sledeće rasporede i komponente:

`TableLayout`

: Pruža dobru kontrolu nad sadržajem ekrana. Taj raspored omogućava razmeštanje komponenata po istom modelu kao HTML oznaka `Table`.

`TableRow`

: Definiše red u prikazu `TableLayout`. Deluje na isti način kao kombinacija HTML oznaka `TR` i `TD`.

`TextView`

: Generiše natpis koji omogućava prikazivanje statičkog teksta na ekranu.

`EditText`

: Generiše polje za tekst u koje se mogu upisivati vrednosti.

`RadioGroup`

: Formira grupu radio-dugmadi.

`RadioButton`

: Generiše radio-dugme.

`Button`

Generiše standardno dugme za pritiskanje.

`View`

Objekat tipa View iskoristićemo za izradu vizuelnog separatora pomoću određenih atributa za visinu i boju.

Trebalo bi da dobro proučite ove komponente jer ćete ih često koristiti u aplikacijama koje pišete. Kada proučavate dokumentaciju određenog rasporeda ili komponente, obratite pažnju na XML atribute. Tako ćete lakše shvatiti vezu između oblika objekta kako je definisan u datoteci sadržaja ekrana *main.xml* i upotrebe u Java kodu (*Tipster.java* i *R.java*) gde se tim objektima pristupa.

U okruženju Eclipse na raspolaganju je i vuzuelni editor ekrana, a postoji i samostalna alatka za izradu grafičkog korisničkog interfejsa koja se zove DroidDraw (*http://www.droiddraw.org/*). Obe alatke omogućavaju izradu ekrana prevlačenjem i spuštanjem komponenata s ponuđene palete, slično svim drugim alatkama za izradu ekranskih obrazaca. Međutim, preporučujem vam da izglede ekrana pravite ručno pomoću XML datoteka, barem dok ste na početku učenja Androida. Kasnije, kada savladate sve detalje XML API-ja za izradu ekrana, taj posao možete prepustiti alatkama za tu namenu.

Datoteka sadržaja ekrana, *main.xml*, sadrži informacije o elementima ekrana (primer 1-6). Komponenta `TableRow` formira jedan red unutar komponente (kontejnera) `TableLayout`. Zadajte onoliko prikaza tipa `TableRow` koliko vam treba redova. U ovom primeru imaćemo osam prikaza `TableRow` – pet za komponente između dugmadi i vizuelnog separatora, i tri u oblasti za rezultate ispod separatora.

Primer 1-6. Datoteka /res/layout/main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Upotreba rasporeda TableLayout da bi se elementi ekrana razmeštali kao u HTML tabeli -->
<TableLayout
        android:id="@+id/TableLayout01"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:stretchColumns="1"
        xmlns:android="http://schemas.android.com/apk/res/android">
    <!-- Red 1: natpis u koloni nula,
         polje za tekst u koloni dva i prošireno na dve kolone.
         To čini ukupno četiri kolone u ovom redu. -->
        <TableRow>
        <TextView
                android:id="@+id/txtLbl1"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_column="0"
                android:text="@string/textLbl1"/>
        <EditText❶
                android:id="@+id/txtAmount"
                android:layout_width="wrap_content"
```

```
            android:layout_height="wrap_content"
            android:numeric="decimal"
            android:layout_column="2"
            android:layout_span="2"
            />
    </TableRow>
 <!-- Red 2: natpis u koloni nula,
      polje za tekst u koloni dva i prošireno na dve kolone.
      To čini ukupno četiri kolone u ovom redu. -->
    <TableRow>
    <TextView
            android:id="@+id/txtLbl2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_column="0"
            android:text="@string/textLbl2"/>
    <EditText
            android:id="@+id/txtPeople"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:numeric="integer"
            android:layout_column="2"
            android:layout_span="2"/>
    </TableRow>
<!-- Red 3: Ovde imamo samo natpis u koloni nula  -->
    <TableRow>
    <TextView
            android:id="@+id/txtLbl3"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/textLbl3"/>
    </TableRow>
<!-- Red 4: Kontejner RadioGroup za komponente RadioButtons u koloni
    Nula, koja je proširena na tri kolone, čime dobijamo po jedno
    radio-dugme po ćeliji u redu tabele. Poslednja ćelija br. 4 sadrži
    polje za tekst u koje se upisuje poseban procenat napojnice -->
    <TableRow>
    <RadioGroup
            android:id="@+id/RadioGroupTips"
            android:orientation="horizontal"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_column="0"
            android:layout_span="3"
            android:checkedButton="@+id/radioFifteen">
            <RadioButton android:id="@+id/radioFifteen"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:text="@string/rdoTxt15"
                    android:textSize="15sp" />
            <RadioButton android:id="@+id/radioTwenty"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:text="@string/rdoTxt20"
                    android:textSize="15sp" />
```

```
            <RadioButton android:id="@+id/radioOther"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:text="@string/rdoTxtOther"
                    android:textSize="15sp" />
        </RadioGroup>
                <EditText
                    android:id="@+id/txtTipOther"
                    android:layout_width="fill_parent"
                    android:layout_height="wrap_content"
                    android:numeric="decimal"/>
        </TableRow>
    <!--  Red za dugmad Calculate i Reset. Dugme Calculate postavljamo
          u kolonu dva, a dugme Reset u kolonu tri. -->
        <TableRow>
        <Button
                android:id="@+id/btnReset"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_column="2"
                android:text="@string/btnReset"/>
        <Button
                android:id="@+id/btnCalculate"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_column="3"
                android:text="@string/btnCalculate"/>
        </TableRow>
     <! – Kontejner TableLayout prihvata umetanje elemenata proizvoljne vrste
          između elemenata TableRow. Zato umećemo prazan prikaz da bismo
          dobili liniju separatora. Taj separatorski prikaz razdvaja oblast
          ispod dugmadi koja će služiti za prikazivanje rezultata
          izračunavanja -->
        <View
                android:layout_height="2px"
                android:background="#DDFFDD"
                android:layout_marginTop="5dip"
                android:layout_marginBottom="5dip"/>

     <! – Još jednom koristimo prikaz TableRow u čiju kolonu nula
          postavljamo natpis, a u kolonu dva polje za rezultate. -->
        <TableRow android:paddingBottom="10dip" android:paddingTop="5dip">
        <TextView
                android:id="@+id/txtLbl4"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_column="0"
                android:text="@string/textLbl4"/>
        <TextView
                android:id="@+id/txtTipAmount"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_column="2"
                android:layout_span="2"/>
```

```
        </TableRow>

        <TableRow android:paddingBottom="10dip" android:paddingTop="5dip">
        <TextView
                android:id="@+id/txtLbl5"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_column="0"
                android:text="@string/textLbl5"/>
        <TextView
                android:id="@+id/txtTotalToPay"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_column="2"
                android:layout_span="2"/>
        </TableRow>

        <TableRow android:paddingBottom="10dip" android:paddingTop="5dip">
        <TextView
                android:id="@+id/txtLbl6"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_column="0"
                android:text="@string/textLbl6"/>
        <TextView
                android:id="@+id/txtTipPerPerson"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_column="2"
                android:layout_span="2"/>
        </TableRow>
    <!--  Kraj svih redova tabele i komponenata u njima -->
</TableLayout>
```

Prikazi TableLayout i TableRow

Ako proučite datoteku *main.xml*, uočićete da je upotreba prikaza `TableLayout` i `TableRow` vrlo jednostavna. Prikaz (kontejner) `TableLayout` zadajete jedanput, a zatim u njega umećete prikaze `TableRow`. U njih možete da umećete komponente svih drugih vrsta, kao što su `TextView`, `EditView` itd.

Dobro proučite atribute, naročito `android:stretchColumns`, `android:layout_column` i `android:layout_span`, koji omogućavaju razmeštanje komponenata na isti način kao što biste to radili u običnoj HTML tabeli. Preporučujem vam da proučite dokumentaciju za te atribute i vidite kako oni deluju u prikazu `TableLayout`.

Kontrolisanje vrednosti koje korisnik unosi

Kontrolisanje ulaznih vrednosti: pogledajte definiciju komponente `EditText` u datoteci *main.xml*, u tački ❶. To je prvo polje za tekst, u koje se upisuje ukupan iznos računa. Želimo da ono dozvoljava upisivanje samo brojeva. Možemo prihvatiti decimalne brojeve pošto u stvarnim restoranima računi mogu biti izraženi u novčanim jedinicama i podjedinicama,

ne samo u osnovnim jedinicama. Zato smo upotrebili atribut `android:numeric` kojem smo zadali vrednost `decimal`. Tako će polje prihvatati celobrojne vrednosti poput 10 i decimalne vrednosti poput 10.12, ali će sprečiti upisivanje drugih vrsta vrednosti.

To je jednostavan i sažet način kontrolisanja ulaznih vrednosti, što će nas poštedeti pisanja obimnog koda za proveru ispravnosti tih vrednosti u datoteci `Tipster.java` i sprečiti da korisnik unosi neprihvatljive vrednosti. Ta Androidova mogućnost definisanja ograničenja u XML obliku veoma je moćna i korisna. Trebalo bi da istražite sve moguće atribute koje data komponenta podržava kako biste izvukli maksimalnu korist od tog skarećnog XML oblika zadavanja ograničenja. Nadam se da će u nekoj budućoj verziji Androida – osim ako mi je potpuno promaklo u tekućoj – moći da se zada opseg za atribut `android:numeric` prihvatljiv opseg vrednosti.

Pošto zadavanje opsega zasad nije moguće (koliko ja znam), u nastavku ćete videti da ipak moramo da eksplicitno ispitamo da li je upisana nula ili vrednost nije ni zadata, kako bismo obezbedili da je aritmetički proračun izvodljiv.

Klasa Tipster.java

Sada ćemo razmotriti klasu *Tipster.java* koja upravlja našom aplikacijom. To je glavna klasa koja definiše ekran aplikacije, obrađuje događaje i sadrži logiku aplikacije.

Dodatak Android za Eclipse formira u projektu datoteku *Tipster.java* iz koje je deo koda prikazan u primeru 1-7.

Primer 1-7. Prvi deo koda klase /src/com/examples/tipcalc/Tipster.java

```
package com.examples.tipcalc;

import android.app.Activity;

public class Tipster extends Activity {
    /** Poziva se pri inicijalzovanju aktivnosti. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

Klasa `Tipster` proširuje klasu `android.app.Activity`. Aktivnost je jedina specijalizovana stvar koju korisnik može da uradi. Klasa `Activity` se stara o formiranju prozora i raspoređivanju komponenata korisničkog interfejsa (GUI) u prozoru. Da biste aktivnost povezali s GUI, treba da pozovete metodu `setContentView(View prikaz)`. Aktivnost možete zamisliti kao okvir koji je prazan i čiju osnovicu popunjavate komponentama GUI.

A sada pogledajte blok koda iz klase *Tipster.java* prikazan u primeru 1-8. Prvo definišemo komponente GUI kao promenljive koje su članovi klase. Pogledajte kao primer tačke ❶ i ❷.

Zatim pozivamo metodu `findViewById(int id)` da bismo dobili identifikatore komponenata. Vrednost identifikatora svake komponente, definisane u datoteci `main.xml`, automatski

se generiše u datoteci `R.java` kada počistite i sklopite projekat u okruženju Eclipse. (Ako ste Eclipse podesili da se projekat sklapa automatski, sadržaj datoteke `R.java` se ažurira kad god nešto izmenite u datoteci `main.xml`.)

Svaka komponenta (prikaz) izvedena je od klase `View` i proizvodi određen element za grafički korisnički interfejs. Tako prikaz tipa `TextView` omogućava prikazivanje statičkog natpisa u GUI, dok komponenta tipa `EditText` proizvodi polje za upisivanje teksta. Pogledajte tačke od ❸ do ❻ u primeru 1-8. Možete videti kako se pomoću metode `findViewById()` dobijaju reference na komponente GUI.

Primer 1-8. Drugi deo koda klase /src/com/examples/tipcalc/Tipster.java

```
public class Tipster extends Activity {
    // Vidžeti u aplikaciji
    private EditText txtAmount;❶
    private EditText txtPeople;
    private EditText txtTipOther;
    private RadioGroup rdoGroupTips;
    private Button btnCalculate;
    private Button btnReset;

    private TextView txtTipAmount;
    private TextView txtTotalToPay;
    private TextView txtTipPerPerson; ❷

    // Za id izabranog radio-dugmeta
    private int radioCheckedId = -1;

    /** Poziva se pri inicijalizovanju aktivnosti. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        // Pojedinim komponentama pristupamo na osnovu njihovih
        // identifikatora iz datoteke R.java
        txtAmount = (EditText) findViewById(R.id.txtAmount);❸
        // Pri pokretanju aplikacije, kursor treba da bude u polju Amount
        txtAmount.requestFocus();❹

        txtPeople = (EditText) findViewById(R.id.txtPeople);
        txtTipOther = (EditText) findViewById(R.id.txtTipOther);

        rdoGroupTips = (RadioGroup) findViewById(R.id.RadioGroupTips);

        btnCalculate = (Button) findViewById(R.id.btnCalculate);
        // Pri pokretanju aplikacije, dugme Calculate je nedostupno
        btnCalculate.setEnabled(false);❺

        btnReset = (Button) findViewById(R.id.btnReset);

        txtTipAmount = (TextView) findViewById(R.id.txtTipAmount);
        txtTotalToPay = (TextView) findViewById(R.id.txtTotalToPay);
        txtTipPerPerson = (TextView) findViewById(R.id.txtTipPerPerson);❻
```

```
// Pri pokretanju aplikacije, polje Other Tip Percentage
// je nedostupno
txtTipOther.setEnabled(false);❼
```

Pitanja upotrebljivosti aplikacije

Trebalo bi da naša aplikacija bude upotrebljiva koliko i svaka druga poznata aplikacija ili veb stranica. Ukratko, ugrađivanje mogućnosti koje proširuju upotrebljivost aplikacije imaće za rezultat dobro iskustvo njenih korisnika. Pogledajte ponovo primer 1-8.

Pogledajte tačku ❹ gde pozivamo metodu `requestFocus()` klase `View`. Pošto je komponenta `EditText` izvedena od klase `View`, ima tu metodu, koju pozivamo zato da nakon pokretanja aplikacije polje za tekst Total Amount dobije fokus i u njega se postavi kursor. To je slično ekranima za prijavljivanje korisnika na popularnim veb lokacijama gde se kursor postavlja u polje za upisivanje imena korisnika.

Sada pogledajte tačku ❺ gde dugme Calculate činimo nedostupnim tako što pozivamo metodu `setEnabled(boolean enabled)` komponente `Button`. To radimo zato da korisnik ne bi mogao da pritisne to dugme pre nego što upiše vrednosti u obavezna polja. Kada bismo korisniku dozvolili da pritisne dugme Calculate a da prethodno ne popuni polja Total Amount (ukupan iznos) i No. of People (broj osoba), morali bismo da napišemo kôd za proveru ispravnosti podataka koji bi prepoznavao takve slučajeve i korisniku prikazivao upozoravajuće poruke da nedostaju vrednosti. Tako dodajemo aplikaciji nepotreban kôd i interakciju s korisnikom. Kada korisnik vidi da je dugme Calculate nedostupno, sasvim je jasno da ne može da izračuna iznos napojnice dok ne upiše sve potrebne vrednosti.

Pogledajte tačku ❼ u primeru 1-8. Tu činimo nedostupnim polje za tekst Other Tip Percentage (drugi procenat napojnice) zato što je radio-dugme „15% tip“ standardno izabrano pri pokretanju aplikacije. Taj podrazumevani izbor pri pokretanju aplikacije zadat je u datoteci *main.xml*. Pogledajte u toj datoteci red gde sledeći iskaz bira radio-dugme „15% tip“:

```
android:checkedButton="@+id/radioFifteen"
```

Atribut `android:checkedButton` kontejnera `RadioGroup` omogućava da jednu od komponenata `RadioButton` u grupi zadate kao standardno izabranu.

Većini korisnika popularnih samostalnih i veb aplikacija poznat je model „nedostupne komponente postaju dostupne kada se ispune određeni uslovi“. Ugrađivanje tih malih pogodnosti čini aplikaciju upotrebljivijom i poboljšava korisnikovo iskustvo pri njenoj upotrebi.

Obrada događaja u korisničkom interfejsu

Kao i popularni Windows, Java Swing, Flex i svaki drugi frejmvork za izradu grafičkog korisničkog interfejsa, Android takođe pruža model događaja koji omogućava obrađivanje određenih događaja u korisničkom interfejsu čiji su izvori akcije korisnika. Sada ćemo videti kako možemo iskoristiti Androidov model događaja u našoj aplikaciji.

Prvo ćemo razmotriti radio-dugmad u korisničkom interfejsu. Potrebno je da znamo koje je radio-dugme korisnik izabrao jer ćemo na osnovu toga odrediti procenat napojnice s kojim ćemo računati. Da bismo „osluškivali“ radio-dugmad, upotrebićemo statički interfejs `OnCheckedChangeListener()` koji će nas obaveštavati kad god se promeni stanje nekog radio--dugmeta.

U našoj aplikaciji želimo da polje Other Tip Percentage bude dostupno samo kada je izabrano radio-dugme Other (drugi). Kada je izabrano dugme „15% tip“ ili „20% tip“, želimo da to polje za tekst bude nedostupno. Osim toga, dodaćemo još logike koja poboljšava upotrebljivost aplikacije. Kao što smo već napomenuli, dugme Calculate ne treba da bude dostupno sve dok sva polja ne budu popunjena ispravnim vrednostima. Preslikano na tri radio-dugmeta, želimo da dugme Calculate bude dostupno samo u sledećim slučajevima:

- Izabrano je radio-dugme Other i polje Other Tip Percentage sadrži ispravnu vrednost.
- Izabrano je radio-dugme „15% tip“ ili „20% tip,“ i oba polja Total Amount i No. of People sadrže ispravne vrednosti.

Pogledajte primer 1-9, koji obrađuje radio-dugmad. Komentari u izvornom kodu objašnjavaju suštinu.

Primer 1-9. Treći deo koda klase /src/com/examples/tipcalc/Tipster.java

```
/*
 * Grupi radio-dugmadi pridružujemo osluškivač OnCheckedChangeListener
 * da bismo znali šta je korisnik izabrao među radio-dugmadima
 */
 rdoGroupTips.setOnCheckedChangeListener(new OnCheckedChangeListener() {

 @Override
 public void onCheckedChanged(RadioGroup group, int checkedId) {
    // Uključivanje/isključivanje polja Other Tip Percentage
   if (checkedId == R.id.radioFifteen
              || checkedId == R.id.radioTwenty) {
       txtTipOther.setEnabled(false);
       /*
        * Dugme Calculate postaje dostupno kada polja Total Amount
        * i No. of People sadrže ispravne vrednosti.
        */
       btnCalculate.setEnabled(txtAmount.getText().length() > 0
                 && txtPeople.getText().length() > 0);
   }
   if (checkedId == R.id.radioOther) {
      // uključujemo polje Other Tip Percentage
      txtTipOther.setEnabled(true);
      // postavljamo fokus u to polje
      txtTipOther.requestFocus();
      /*
       * Dugme Calculate postaje dostupno kada polja Total Amount i
       * No. of People sadrže ispravne vrednosti. Osim toga, proveravamo
       * i da li je korisnik popunio polje Other Tip Percentage
       * pre nego što uključimo dugme Calculate.
       */
```

```
      btnCalculate.setEnabled(txtAmount.getText().length() > 0
              && txtPeople.getText().length() > 0
              && txtTipOther.getText().length() > 0);
    }
    // Da bismo odredili koju je opciju procenta napojnice korisnik izabrao
    radioCheckedId = checkedId;
  }
});
```

Obrada događaja upisivanja znakova u polja za tekst

Kao što sam već napomenuo, dugme Calculate ne sme biti dostupno pre nego što polja za tekst sadrže ispravne vrednosti. Dakle, moramo obezbediti da dugme Calculate postane dostupno tek pošto korisnik popuni ispravnim vrednostima polja Total Amount, No. of People i Other Tip Percentage. Polje Other Tip Percentage je dostupno samo kada je izabrano radio-dugme Other Tip Percentage.

Ne moramo da brinemo o tipovima vrednosti u polju, odnosno da li je korisnik upisao negativne vrednosti ili slova zato što je atribut `android:numeric` definisan za prikaze tipa `EditText` (polje za tekst), čime se ograničava tip vrednosti koje korisnik može da upiše u polje. Moramo obezbediti da su polja popunjena.

U tu svrhu upotrebićemo statički interfejs `OnKeyListener()` koji će nas obavestiti da je pritisnut neki taster. Obaveštenje stiže u aplikaciju pre nego što se kôd pritisnutog tastera pošalje komponenti `EditText`.

Pogledajte kôd u primerima 1-10 i 1-11, koji obrađuje događaje pritiskanja tastera u poljima za tekst. Kao i u primeru 1-9, komentari u izvornom kodu objašnjavaju suštinu.

Primer 1-10. Četvrti deo koda klase /src/com/examples/tipcalc/Tipster.java

```
/*
 * Poljima Tip Amount, No. of People i Other Tip Percentage
 * pridružujemo osluškivač KeyListener
 */
txtAmount.setOnKeyListener(mKeyListener);
txtPeople.setOnKeyListener(mKeyListener);
txtTipOther.setOnKeyListener(mKeyListener);
```

Obratite pažnju na to da imamo samo jedan osluškivač umesto anonimnih/internih osluškivača za svako polje pojedinačno. Nisam siguran da li je moj stil bolji ili preporučljiv, ali ja uvek pišem u ovom stilu ako osluškivači treba da obavljaju određene akcije koje su im svima zajedničke. U ovom slučaju, svim poljima za tekst zajedničko je to što ne smeju biti prazna i samo ako su sva popunjena, dugme Calculate sme biti dostupno.

Primer 1-11. Peti deo koda za osluškivač KeyListener.java

```
/*
 * KeyListener pridružen poljima Total Amount, No of People i Other Tip
 * Percentage. Ovaj osluškivač treba da obezbedi ispunjavanje sledećih
 * uslova:
 *
 * 1) Ako korisnik izabere opciju Other Tip Percentage, polje Other Tip Percentage
```

```
 * treba da sadrži ispravan procenat napojnice. Dugme Calculate može biti dostupno
 * samo ako je korisnik upisao ispravnu vrednost.
 *
 * 2) Ako korisnik ne popuni polja Total Amount i No. of People, ne možemo da izračunamo
 * iznos. Zato dugme Calculate oslobađamo samo kada korisnik upiše ispravne vrednosti.
 */
private OnKeyListener mKeyListener = new OnKeyListener() {
    @Override
    public boolean onKey(View v, int keyCode, KeyEvent event) {

    switch (v.getId()) { ❶
    case R.id.txtAmount: ❷
    case R.id.txtPeople: ❸
        btnCalculate.setEnabled(txtAmount.getText().length() > 0
                && txtPeople.getText().length() > 0);
        break;
    case R.id.txtTipOther: ❹
        btnCalculate.setEnabled(txtAmount.getText().length() > 0
                && txtPeople.getText().length() > 0
                && txtTipOther.getText().length() > 0);
        break;
    }
    return false;
    }

};
```

U tački ❶ u primeru 1-11, ispitujemo vrednost identifikatora prikaza. Imajte u vidu da je svakoj komponenti definisanoj u datoteci *main.xml* pridružen jedinstven identifikator. Vrednosti tih identifikatora definisane su u generisanoj klasi `R.java`.

U tačkama ❷ i ❸, ako je izvor događaja pritiskanja tastera polje Total Amount ili No. of People, ispitujemo vrednost unetu u polje kako bismo se uverili da korisnik nije ostavio oba polja prazna.

U tački ❹ utvrđujemo da li je korisnik izabrao radio-dugme Other, a zatim proveravamo da li je polje za tekst Other prazno. Osim toga, još jedanput proveravamo da li su prazna polja Total Amount i No. of People.

Svrha našeg osluškivača `KeyListener` sada je jasna: obezbediti da su sva polja za tekst popunjena i samo u tom slučaju osloboditi dugme Calculate.

Obrada događaja pritiskanja dugmadi

Sada ćemo razmotriti dugmad Calculate i Reset. Kada korisnik pritisne jedno od njih, statički interfejs `OnClickListener()` omogućava nam da saznamo koje je dugme pritisnuto.

Isto kao što smo uradili za polja za tekst, pravimo samo jedan osluškivač i pomoću njega otkrivamo koje je dugme pritisnuto. U zavisnosti od pritisnutog dugmeta, pozivamo metodu `calculate()` ili `reset()`.

Primer 1-12 prikazuje kako se osluškivač pridružuje dugmadima.

Primer 1-12. Šesti deo koda klase /src/com/examples/tipcalc/Tipster.java

```
/* Pridružuje osluškivač dugmadima Calculate i Reset */
btnCalculate.setOnClickListener(mClickListener);
btnReset.setOnClickListener(mClickListener);
```

Primer 1-13 prikazuje kako se na osnovu vrednosti identifikatora prikaza koji je primio obaveštenje o događaju utvrđuje koje je dugme pritisnuto.

Primer 1-13. Sedmi deo koda klase /src/com/examples/tipcalc/Tipster.java

```
/**
 * ClickListener za dugmad Calculate i Reset.
 * U zavisnosti od pritisnutog dugmeta, poziva se
 * odgovarajuća metoda.
 */
private OnClickListener mClickListener = new OnClickListener() {

    @Override
    public void onClick(View v) {
        if (v.getId() == R.id.btnCalculate) {
            calculate();
        } else {
            reset();
        }
    }
};
```

Resetovanje aplikacije

Kada korisnik pritisne dugme Reset, polja za tekst treba isprazniti, izabrati podrazumevano radio-dugme „15% tip“ i ako ih ima, isprazniti prethodno izračunate rezultate.

Primer 1-14 prikazuje metodu `reset()`.

Primer 1-14. Osmi deo koda klase /src/com/examples/tipcalc/Tipster.java

```
/**
 * Prazni polja za rezultate pri dnu ekrana, kao i ostala polja za tekst
 * i bira podrazumevani procenat.
 */
private void reset() {
    txtTipAmount.setText("");
    txtTotalToPay.setText("");
    txtTipPerPerson.setText("");
    txtAmount.setText("");
    txtPeople.setText("");
    txtTipOther.setText("");
    rdoGroupTips.clearCheck();
    rdoGroupTips.check(R.id.radioFifteen);
    // set focus on the first field
    txtAmount.requestFocus();
}
```

Provera ispravnosti ulaznih podataka za izračunavanje napojnice

Kao što sam već napomenuo, u aplikaciji ograničavamo tip vrednosti koje korisnik može da upiše u polja za tekst. Međutim, korisnik i dalje može da upiše nule u polja Total Amount, No. of People i Other Tip Percentage, što će pri izračunavanju napojnice prouzrokovati grešku deljenja nulom.

Ako korisnik upiše nulu, moramo prikazati poruku da treba da se upisuju vrednosti različite od nule. To rešavamo pomoću metode `showErrorAlert(String errorMessage, final int fieldId)`, koju ćemo objasniti u nastavku teksta.

Prvo pogledajte primer 1-15 koji prikazuje metodu calculate(). Obratite pažnju na to kako se vrednosti koje je korisnik upisao preslikavaju u vrednosti tipa double.

Pogledajte tačke ❶ i ❷ gde ispitujemo da li su unete nule. Ako korisnik upiše nulu, prikazujemo upozorenje. Dalje, pogledajte tačku ❸, gde oslobađamo polje Other Tip Percentage zato što je korisnik izabrao radio-dugme Other. I u ovom slučaju moramo proveriti da li je kao procenat napojnice upisana nula.

Pri pokretanju aplikacije, radio-dugme „15% tip“ postavlja se u izabrano stanje. Ako zatim korisnik izabere drugo radio-dugme, identifikator izabranog dugmeta dodeljujemo internoj promenljivoj `radioCheckedId`, kao i u primeru 1-9, u kodu osluškivača `OnCheckedChangeListener`.

Ali ukoliko korisnik prihvati podrazumevani početni izbor, promenljiva `radioCheckedId` imaće podrazumevanu vrednost –1. Ukratko, nećemo znati koje je radio-dugme izabrano. Razume se, pošto znamo koje je dugme podrazumevani izbor, mogli smo da kodiramo nešto drugačiju logiku, u kojoj se podrazumeva da je procenat 15% ako promenljiva `radioCheckedId` ima vrednost –1. Ali ako pogledate dokumentaciju za API, videćete da možemo pozvati metodu `getCheckedRadioButtonId()` kontejnera `RadioGroup` ali ne i za pojedinu radio-dugmad. Razlog je to što nam osluškivač `OnCheckedChangeListener` stavlja na raspolaganje identifikator izabranog radio-dugmeta.

Prikazivanje rezultata

Iznos napojnice se izračunava vrlo jednostavno. Ako su ulazni podaci ispravni, indikator tipa boolean `isError` ima vrednost false. U tačkama ❹ i ❺ primera 1-15 obavljamo jednostavan obračun napojnice. Dalje, izračunate vrednosti upisujemo u polja `TextView`, u tačkama ❻ i ❼.

Primer 1-15. Deveti deo koda klase /src/com/examples/tipcalc/Tipster.java.

```
/**
 * Obračun napojnice na osnovu podataka koje je korisnik zadao.
 */
private void calculate() {
    Double billAmount = Double.parseDouble(
        txtAmount.getText().toString());
    Double totalPeople = Double.parseDouble(
        txtPeople.getText().toString());
```

```
    Double percentage = null;
    boolean isError = false;
    if (billAmount < 1.0) {❶
        showErrorAlert("Enter a valid Total Amount.",
            txtAmount.getId());
        isError = true;
    }

    if (totalPeople < 1.0) {❷
        showErrorAlert("Enter a valid value for No. of People.",
            txtPeople.getId());
        isError = true;
    }

    /*
     * Ako korisnik nije izabrao nijedno radio-dugme, to znači da važi
     * podrazumevani procenat od 15%. Ali je ipak sigurnije da
     * proverimo da li je tako.
     */
    if (radioCheckedId == -1) {
        radioCheckedId = rdoGroupTips.getCheckedRadioButtonId();
    }
    if (radioCheckedId == R.id.radioFifteen) {
        percentage = 15.00;
    } else if (radioCheckedId == R.id.radioTwenty) {
        percentage = 20.00;
    } else if (radioCheckedId == R.id.radioOther) {
        percentage = Double.parseDouble(
            txtTipOther.getText().toString());
        if (percentage < 1.0) {❸
            showErrorAlert("Enter a valid Tip percentage",
                txtTipOther.getId());
            isError = true;
        }
    }
    /*
     * Ako su polja popunjena ispravnim vrednostima, prelazimo
     * na obračun napojnice
     */
    if (!isError) {
        Double tipAmount = ((billAmount * percentage) / 100); ❹
        Double totalToPay = billAmount + tipAmount;
        Double perPersonPays = totalToPay / totalPeople;❺

        txtTipAmount.setText(tipAmount.toString());❻
        txtTotalToPay.setText(totalToPay.toString());
        txtTipPerPerson.setText(perPersonPays.toString());❼
    }
}
```

Prikazivanje upozorenja

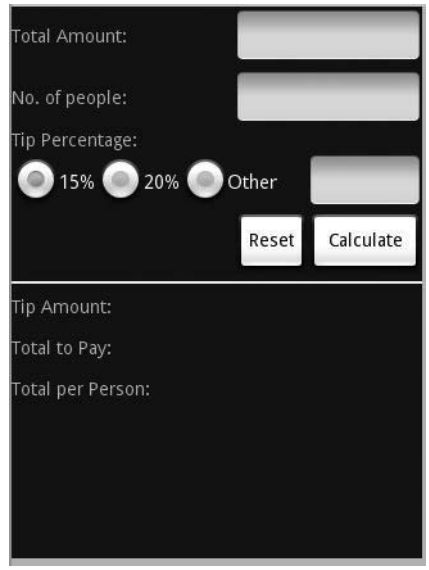
U Androidu postoji klasa `AlertDialog` koja omogućava prikazivanje iskačućih poruka. Može se prikazati dijalog s tekstom poruke i najviše tri dugmeta.

Primer 1-16 prikazuje metodu `showErrorAlert` koja pomoću objekta tipa `AlertDialog` prikazuje poruke o greškama. Obratite pažnju na to da se metodi prosleđuju dva argumenta: `String errorMessage` i `int fieldId`. Prvi argument je tekst poruke o grešci koji želimo da prikažemo korisniku. Argument `fieldId` je identifikator polja u kojem je nastalo stanje greške. Pošto korisnik ukloni dijalog s porukom, ta vrednost `fieldId` će nam omogućiti da premestimo fokus na to polje kako bi korisnik znao u kojem je polju greška.

Primer 1-16. Deseti deo koda klase /src/com/examples/tipcalc/Tipster.java

```
/**
 * Prikazuje poruku o grešci u iskačućem dijalogu
 *
 * @param errorMessage
 *            tekst poruke koja se prikazuje
 * @param fieldId
 *            identifikator polja u kojem je došlo do greške.
 *            Potreban nam je da bismo fokus postavili
 *            na to polje posle uklanjanja dijaloga
 */
private void showErrorAlert(String errorMessage,
    final int fieldId) {
    new AlertDialog.Builder(this).setTitle("Error")
    .setMessage(errorMessage).setNeutralButton("Close",
            new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog,
                        int which) {
                    findViewById(fieldId).requestFocus();
                }
            }).show();
}
```

Kada sve sklopite u jednu celinu, trebalo da izgleda kao na slici 1-39.



Slika 1-39. Program Tipster u akciji

Zaključak

Razvijanje aplikacija za OS Android ne razlikuje se značajno od razvijanja aplikacija s drugim kompletima za GUI, kao što su Microsoftov Windows, X Windows, Java Swing ili Adobe Flex. Razume se, Android ima svoje posebnosti – ukupno gledano – veoma dobar dizajn. Model sadržaja ekrana definisanog u XML obliku veoma je pogodan i upotrebljiv za izradu složenih korisničkih interfejsa pomoću jednostanog XML koda. Osim toga, model obrade događaja je jednostavan, bogat mogućnostima i intuitivan za upotrebu u kodu.

URL za preuzimanje izvornog koda

Izvorni kôd primera Tipster možete preuzeti sa adrese *https://github.com/androidcook/Android-Cookbook-Examples*.