BRZI ODGOVORI NA UOBIČAJENE PROBLEME

Android Studio IDE
kuvar za razvoj aplikacija

Više od 100 recepata za rešavanje svakodnevnih programerskih problema

Rick Boyer **Kyle Mew**

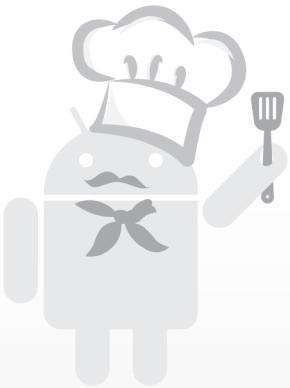
kompjuter biblioteka [PACKT] PUBLISHING

BRZI ODGOVORI NA UOBIČAJENE PROBLEME

Android Studio IDE
kuvar za razvoj aplikacija

Više od 100 recepata za rešavanje svakodnevnih programerskih problema

Rick Boyer **Kyle Mew**



kompjuter biblioteka

[PACKT] PUBLISHING
BIRMINGHAM - MUMBAI

Izdavač:
kompjuter biblioteka

Obalskih radnika 15, Beograd

Tel: 011/2520272

e-mail: kombib@gmail.com

internet: www.kombib.rs

Urednik: Mihailo J. Šolajić

Za izdavača, direktor:
Mihailo J. Šolajić

Autori: Rick Boyer i Kyle Mew

Prevod: Slavica Prudkov

Lektura: Miloš Jevtović

Slog : Zvonko Aleksić

Znak Kompjuter biblioteke:
Miloš Milosavljević

Štampa: „Pekograf“, Zemun

Tiraž: 500

Godina izdanja: 2016.

Broj knjige: 486

Izdanje: Prvo

ISBN: 978-86-7310-509-3

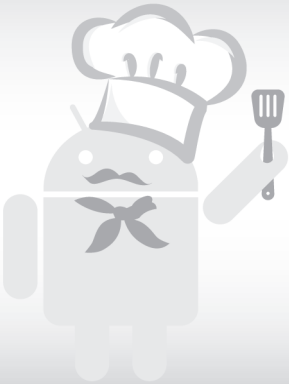
Android Application Development Cookbook

Second Edition

by Rick Boyer and Kyle Mew

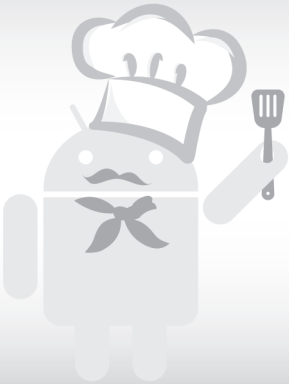
ISBN 978-1-78588-619-5

CIP - Каталогизација у публикацији
Народна библиотека Србије, Београд,
се добија на захтев



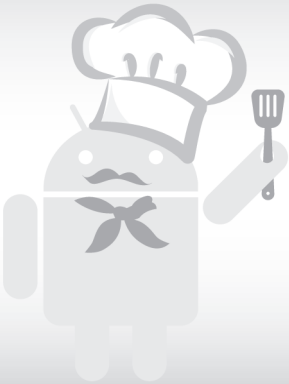
Kratak sadržaj

Contents



Sadržaj

POGLAVLJE 15



UVOD

Android je prvi put izdat 2007. godine, nakon što ga je kupio „Google, Inc.“. Na početku je korišćen primarno za mobilne uređaje. Android 3.0 je dodao funkcije koje su mogle da se iskoriste za rastuće tržište tableta.

„Google“ je 2014. godine objavio da Android ima više od jedne milijarde aktivnih korisnika! S obzirom da je na Google Playu dostupno više od milion aplikacija, sada je najuzbudljivije vreme za priključivanje Android zajednici!

Izdata na početku 2016. godine, verzija Androida 6.0 sadrži uzbudljive nove funkcije, kako za korisnike, tako i za programere.

ŠTA OBUHVATA OVA KNJIGA?

U Poglavlju 1, „*Aktivnosti*“, opisane su aktivnosti koje predstavljaju osnovne gradivne blokove za većinu aplikacija. Videćete primere uobičajenih zadataka, kao što su kreiranje aktivnosti i prosleđivanje kontrole iz jedne aktivnosti u drugu.

U Poglavlju 2, „*Rasporedi*“, govori se o opcijama Layout. Iako su aktivnosti važne za korisnički interfejs, rasporedi, u stvari, definišu šta korisnik vidi na ekranu. Naučićete osnovne opcije rasporeda koje su dostupne i najbolja praktična rešenja za njihovu upotrebu.

U Poglavlju 3, „*Prikazi, vidžeti i stilovi*“, istražujemo osnovne objekte korisničkog interfejsa, od kojih su izgrađeni svi rasporedi. Vidžeti uključuju, sve od dugmadi i tekstualnih polja, do komplikovanijih okvira za dijalog NumberPicker i Calendar.

U Poglavlju 4, „*Meniji*“, saznaćete kako se upotrebljavaju meniji u Android sistemu. Naučićete kako da kreirate menije i kako da kontrolišete njihovo ponašanje pri pokretanju.

U Poglavlju 5, „*Istraživanje fragmenata, AppWidgetsa i sistemskog korisničkog menija*“, prikazujemo kako se kreiraju fleksibilniji korisnički interfejsi ponovnom upotrebom komponenata korisničkog interfejsa pomoću Fragmenta. Iskoristićete nove funkcije operativnog sistema sa providnim sistemskim linijama ili ćete učiniti da sistemski korisnički interfejs u potpunosti nestane pomoću Immersive Modea.

Poglavlje 6, „*Upotreba podataka*“, pomaže da otkrijete više metoda koje Android nudi za zadržavanje podataka i da saznate kada je neku opciju najbolje upotrebiti. Primer klase Loadera prikazuje efikasno rešenje za predstavljanje podataka bez vezivanja UI Threada.

U Poglavlju 7, „*Upozorenja i obaveštenja*“, prikazujemo više opcija za prikaz obaveštenja za korisnike. Postojeće opcije su upozorenja za aplikacije, opcije za upotrebu sistemskih obaveštenja i Heads Up obaveštenja.

Poglavlje 8, „*Upotreba ekrana osetljivog na dodir i senzora*“, pomaže da naučite događaje za rukovanje standardnim korisničkim interakcijama, kao što su klik ne dugme, dug pritisak i gestovi. Pristupite hardverskim senzorima uređaja za određivanje promena orijentacije i pokreta uređaja i upotrebu kompasa.

Poglavlje 9, „*Grafike i animacija*“, pomaže da „oživite“ aplikacije pomoću animacija! Iskoristite mnoge od opcija koje Android nudi za kreiranje animacija – za jednostavne bitmape i prilagođene svojstvene animacije.

U Poglavlju 10, „*Prvi pogled na OpenGL ES*“, opisan je OpenGL. Kada su vam potrebne 2D i 3D grafike visoke performanse, upotrebite Open Graphics biblioteku. Android podržava OpenGL, međuplatformski Graphics API.

U Poglavlju 11, „*Multimedija*“, naučićete kako da iskoristite hardverske funkcije za puštanje audio snimaka. Upotrebite Android Intents da biste pozvali standardnu aplikaciju kamere ili detaljnije pregledajte API-je kamere da biste direktno kontrolisali kameru.

U Poglavlju 12, „*Telefonija, mreže i Web*“, koristimo funkcije Telephony za pokretanje telefonskog poziva i slušanje ulaznih telefonskih događaja. Videćete kako možete da šaljete i primate SMS (tekstualne) poruke. Upotrebite WebView u aplikaciji da biste prikazali web stranice i naučili kako da upotrebite Volley za direktnu komunikaciju sa web servisima.

U Poglavlju 13, „*Utvrđivanje lokacije i geografskih zona*“, prikazano je kako možete da odredite lokaciju korisnika i primenite najbolja rešenja da aplikacija ne istroši bateriju. Upotrebite nove Location API-je za primanje ažuriranih podataka o lokaciji i kreiranje Geofencesa.

Poglavlje 14, „*Završavanje aplikacije koja je spremna za Play Store*“, pomaže da uredite aplikaciju za Play Store i naučite kako da implementirate naprednije funkcije, kao što su alarm i AsyncTask za pozadinsku obradu. Videćete kako možete da dodate Google Cloud Messaging (obaveštenja) u aplikaciju i iskoristite Google Sign-in.

U Poglavlju 15, „*Pozadinski prikaz kao opcija servisa*“, istražujemo šta može pozadinski prikaz kao provajder servisa da ponudi za određene aplikacije. Uporedite nekoliko najpopularnijih provajdera koji nude izvornu Android podršku i opcije za besplatnu prijavu.

ŠTA VAM JE POTREBNO ZA OVU KNJIGU?

Razvijanje Android aplikacija zahteva Android SDK koji je dostupan na više platformi, uključujući Windows, Mac i Linux.

Iako nije potrebno, u ovoj knjizi ćemo koristiti Android Studio, zvanični Android IDE. Ako ste novi Android programer, posetite sledeći link da biste pregledali aktuelne sistemske zahteve i preuzeli Android Studio, zajedno sa SDK paketom za vašu platformu:

http://developer.android.com/sdk/index.html

Android SDK i Android Studio su besplatni.

ZA KOGA JE OVA KNJIGA?

U ovoj knjizi pretpostavlja se da imate osnovno znanje o konceptima programiranja i Android osnovama. Ako ste, pak, novi korisnik Androida i želite da naučite više, tako što ćete direktno preći na kod, znajte da ova knjiga pruža širok raspon uobičajenih zadataka.

Pošto je ovo „kuvar“, možete da preskočite na temu koja vas interesuje i upotrebite kod u aplikaciji što je brže moguće.

ODELJCI

U ovoj knjizi pronaći ćete nekoliko naslova koji se često pojavljuju (Priprema, Kako da uradite..., Kako funkcioniše..., Postoji i više... i Takođe vidite).

Da bismo vam pružili jasne instrukcije kako da uradite jedan recept, upotrebićemo sledeće odeljke:

PRIPREMA

Ovaj odeljak ukazuje šta da očekujete u receptu i kako da podesite softver ili bilo koje preliminarne opcije koje su potrebne za recept.

Kako da uradite...

Ovaj odeljak sadrži korake koji su potrebni da biste pratili recept.

Kako funkcioniše...

Ovaj odeljak se, obično, sastoji od detaljnog objašnjenja onoga što se dešava u prethodnom odeljku.

Postoji i više...

Ovaj odeljak se sastoji od dodatnih informacija o receptu da bi čitaoci saznali više o receptu.

Takođe vidite

Ovaj odeljak pruža korisne linkove ka drugim korisnim informacijama u vezi recepta.

KONVENCIJE

U ovoj knjizi pronaći ćete više različitih stilova za tekst koje sam upotrebio za različite vrste informacija. Evo nekih primera ovih stilova i objašnjenja njihovog značenja.

Reči koda u tekstu, nazivi tabela baze podataka, nazivi direktorijuma, nazivi fajlova, ekstenzije fajla, nazivi putanja, kratki URL-ovi i korisnički unos su prikazani na sledeći način: „Zahtevanje JSON odgovora pomoću funkcije `JsonObjectRequest`() obično funkcioniše isto kao i funkcija `StringRequest`()“.

Blok koda je postavljen na sledeći način:

```
<activity
    android:name=".MainActivity"
    android:label="@string/app_name" >
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name=
            "android.intent.category.LAUNCHER"/>
    </intent-filter>
</activity>
```

Novi termini i važne reči su napisane masnim slovima. Reči koje vidite na ekranu - na primer, u menijima ili okvirima za dijalog, biće prikazane u tekstu na sledeći način: „Upotrebite standardnu Phone & Tablet opciju i izaberite opciju Empty Activity kada se zatraži Activity Type“.

Upozorenja ili važne napomene su prikazani u ovakvom okviru.

Saveti i trikovi su prikazani ovako.

POVRATNE INFORMACIJE ČITALACA

Povratne informacije od naših čitalaca su uvek dobrodošle. Obavestite nas šta mislite o ovoj knjizi – šta vam se dopalo ili šta vam se možda nije dopalo. Povratne informacije čitalaca su nam važne da bismo ubuduće kreirali naslove od kojih ćete dobiti maksimum.

Da biste nam poslali povratne informacije, jednostavno nam pošaljite e-mail na adresu feedback@packtpub.com i u naslovu poruke napišite naslov knjige.

Ako postoji tema za koju ste specijalizovani i zainteresovani ste da pišete ili sarađujete na nekoj od knjiga, pogledajte vodič za autore na adresi www.packtpub.com/authors.

KORISNIČKA PODRŠKA

Sada, kada ste ponosni vlasnik „Packt“ knjige, mi imamo mnogo štošta da vam ponudimo da bismo vam pomogli da izvučete maksimum iz svoje narudžbine.

Preuzimanje primera koda

Možete da preuzmete fajlove sa primerima koda za ovu knjigu sa vašeg naloga na adresi http://www.packtpub.com. Ako ste ovu knjigu kupili na drugom mestu, možete da posetite stranicu http://www.packtpub.com/support i da se registrujete da biste e-mailom dobili fajlove.

Možete da preuzmete fajlove sa primerima koda, prateći sledeće korake:

1. Prijavite se ili se registrujte na našem web sajtu, koristeći svoju e-mail adresu i lozinku.
2. Postavite kursor na karticu **SUPPORT** na vrhu stranice.
3. Kliknite na **Code Downloads & Errata**.
4. U polje **Search** unesite naslov knjige.
5. Izaberite knjigu za koju želite da preuzmete fajlove sa primerima koda.
6. Iz padajućeg menija izaberite mesto na kojem ste kupili knjigu.
7. Kliknite na **Code Download**.

Kada su fajlovi preuzeti, ekstrahujte direktorijum, koristeći najnoviju verziju:

- WinRAR / 7-Zip za Windows
- Zipeg / iZip / UnRarX za Mac
- 7-Zip / PeaZip za Linux

Štamparske greške

Iako smo preduzeli sve mere da bismo obezbedili tačnost sadržaja, greške se dešavaju. Ako pronađete grešku u nekoj od naših knjiga (u tekstu ili u kodu), bili bismo zahvalni ako biste nam to prijavili. Time možete da poštedite čitaoce od frustracije i nama da pomognete da poboljšamo naredne verzije ove knjige. Ako pronađete neku štamparsku grešku, molimo vas da nas obavestite, tako što ćete posetiti stranicu http://www.packtpub.com/submit-errata, selektovati knjigu, kliknuti na link **Errata Submission Form** i uneti detalje o grešci koju ste pronašli. Kada je greška verifikovana, vaša prijava će biti prihvaćena i greška će biti aploudovana na naš web sajt ili dodata u listu postojećih grešaka, pod odeljkom Errata za određeni naslov.

Da biste pregledali prethodno prijavljene greške, posetite stranicu https://www.packtpub.com/books/content/support i unesite naslov knjige u polje za pretragu. Tražena informacija će biti prikazana u odeljku **Errata**.

Piraterija

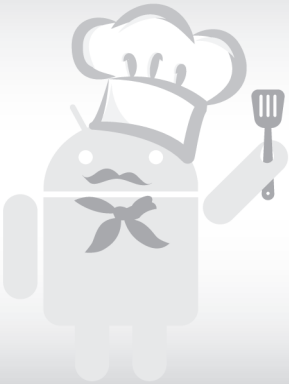
Piraterija autorskog materijala na Internetu je aktuelan problem na svim medijima. Mi u „Packtu“ zaštitu autorskih prava i licenci shvatamo veoma ozbiljno. Ako pronađete ilegalnu kopiju naših knjiga, u bilo kojoj formi, na Internetu, molimo vas da nas o tome obavestite i da nam pošaljete adresu lokacije ili naziv web sajta da bismo mogli da podnesemo tužbu.

Molimo vas, kontaktirajte sa nama na adresi copyright@packtpub.com i pošaljite nam link ka sumnjivom materijalu.

Zahvalni smo vam na pomoći u zaštiti naših autora i mogućnosti da vam pružimo vredan sadržaj.

Pitanja

Možete da sa nama kontaktirate na adresi questions@packtpub.com ako imate bilo kakvih problema sa bilo kojim aspektom knjige i mi ćemo učiniti sve što je u našoj moći da te probleme rešimo.



1

Aktivnosti

Ovo poglavlje obuhvata sledeće recepte:

- deklarisanje aktivnosti
- pokretanje nove aktivnosti pomoću ciljnog objekta
- prekopčavanje između aktivnosti
- prosleđivanje podataka drugoj aktivnosti
- vraćanje rezultata iz aktivnosti
- snimanje statusa aktivnosti
- čuvanje podataka trajne aktivnosti
- razumevanje „životnog ciklusa“ aktivnosti

UVOD

Android SDK obezbeđuje moćnu alatku za programiranje mobilnih uređaja; najbolji način da ovladate takvom alatkom je da odmah počnete da je koristite. Iako možete da pročitate ovu knjigu od početka do kraja, specifično je dizajnirana da omogući da preskočite na specifične zadatke i odmah dobijete rezultate.

Aktivnosti su osnovni gradivni blokovi većine Android aplikacija, pošto klasa aktivnosti obezbeđuje interfejs između aplikacije i ekrana. Većina Android aplikacija će imati bar jednu aktivnost, ako ne i više (ali one nisu potrebne). Pozadinske servisne aplikacije neće obavezno zahtevati aktivnost ako ne postoji korisnički interfejs.

U ovom poglavlju objasnićemo kako se *deklarišu* i *pokreću* aktivnosti unutar aplikacije i kako se upravlja nekim aktivnostima odjednom deljenjem podataka između njih, zahtevanjem rezultata iz njih i pozivanjem jedne aktivnosti iz druge.

Takođe ćemo ukratko istražiti **ciljni** objekat, koji se često koristi zajedno sa aktivnostima. Ciljni objekti mogu da se koriste za prenos podataka između aktivnosti u aplikaciji, kao i u eksternim aplikacijama, kao što su one koje su uključene u Android operativni sistem (uobičajeni primer je upotreba ciljnog objekta za pokretanje standardnog web pretraživača).

Na početku razvijanja Android aplikacije treba da otvorite Android Studio stranicu da biste preuzeli novi Android Studio IDE i Android SDK paket: http://developer.android.com/sdk/index.html

DEKLARISANJE AKTIVNOSTI

Aktivnosti i druge komponente aplikacije, kao što su **servisi**, deklarisani su u XML fajlu koji se zove `AndroidManifest`. Deklarisanje aktivnosti je način na koji govorimo sistemu o našoj aktivnosti. Na primer, aplikacija će obično ukazati da bi trebalo da bude vidljiva bar jedna aktivnost kao desktop ikonica koja će služiti kao glavna ulazna tačka u aplikaciju.

Priprema

Android Studio je nova alatka koja se koristi za razvoj Android aplikacija, zamenjujući sada već zastarelo **Eclipse ADT** rešenje. Android Studio ćemo upotrebiti za sve recepte koji su prikazani u ovoj knjizi, pa, stoga, ako ga još niste instalirali, posetite web sajt Android Studioa (link je ranije prikazan) da biste instalirali IDE i SDK paket.

Kako da uradite...

Za ovaj prvi primer mi ćemo vas provesti kroz proces kreiranja novog projekta. Android Studio obezbeđuje **Quick Start** wizard koji će vam veoma olakšati posao. Pratite sledeće korake:

1. Pokrenite Android Studio i otvoriće se okvir za dijalog **Welcome to Android Studio**.
2. Kliknite na opciju **Start a new Android Studio project**.
3. Unesite naziv aplikacije; za ovaj primer mi ćemo upotrebiti **DeclareAnActivity**. Kliknite na **Next**.
4. U okviru za dijalog **Add an** `Activity` **to Mobile** kliknite na dugme **Blank** `Activity`, a zatim kliknite na **Next**.
5. U okviru za dijalog Target Android Devices izaberite opciju **Android 6.0 (API 23)** kao minimalni SDK (za ovaj primer stvarno nije važno koji nivo API-ja izaberete, pošto aktivnosti postoje od API-ja nivoa 1, ali biranje novijeg izdanja se smatra najboljom praksom). Kliknite na **Next**.

6. Pošto smo ranije izabrali **Blank Activity** opciju, prikazan je okvir za dijalog **Customize the Activity**. Možete da ostavite standardna podešavanja, ali ne zaboravite da je standardni naziv aktivnosti `MainActivity`. Kliknite na **Finish**.

Nakon zatvaranja wizarda, Android Studio će kreirati fajlove projekta. Za ovaj recept kreiraće dva fajla koja ćemo ispitati, a to su `MainActivity.java` (koji odgovara nazivu aktivnosti koji je pomenut u koraku 6) i `AndroidManifest.xml`.

Fajl `MainActivity.java` je prilično jednostavan i sadrži osnove. Razlog je činjenica da smo izabrali opciju **Blank** `Activity` (u koraku 4). Sada pogledajte fajl `AndroidManifest.xml`. Ovde ćemo, u stvari, da deklarišemo aktivnost. Unutar elementa `<application>` nalazi se element `<activity>`:

```
<activity
    android:name=".MainActivity"
    android:label="@string/app_name">
    <intent-filter>
        <action android:name="android.intent.action.MAIN"/>

        <category android:name=
            "android.intent.category.LAUNCHER"/>
    </intent-filter>
</activity>
```

Kada pregledate ovaj xml unutar Android Studioa, možda ćete primetiti da element oznake prikazuje tekst, kao što je definisano u izvornom fajlu strings.xml. Ovo je samo mali primer poboljšanja u novom IDE-u.

Kako funkcioniše...

Deklarisanje aktivnosti je, u stvari, jednostavno deklarisanje elementa `<activity>` i određivanje naziva klase aktivnosti pomoću atributa android:`name`. Dodavanjem elementa `<activity>` u Android Manifest specifikujemo svoju nameru da uključimo ovu komponentu unutar aplikacije. Bilo koje aktivnosti (ili bilo koje druge komponente) koje nisu deklarisane u manifestu neće biti uključene u aplikaciju. Pokušaj pristupa ili upotrebe nedeklarisane komponente će rezultirati generisanjem izuzetka pri pokretanju.

U prethodnom kodu postoji još jedan atribut - android:label. On ukazuje na naslov koji je prikazan na ekranu i na oznaku ikonice (ako je ovo Launcher aktivnost).

Za kompletnu listu dostupnih atributa aplikacije pogledajte sledeći izvor: http://developer.android.com/guide/topics/manifest/activity-element.html.

POKRETANJE NOVE AKTIVNOSTI POMOĆU INTENT OBJEKTA

Model Android aplikacije može se smatrati servisno-orijentisanim, sa aktivnostima kao komponentama i intentom kao porukama koje su poslate između njih. Ovde se intent koristi za pokretanje aktivnosti koja prikazuje evidenciju poziva korisnika, ali intenti mogu da se upotrebe i za mnoge druge zadatke, o kojima će biti reči kasnije u ovoj knjizi.

Priprema

Da bismo objašnjenje zadržali jednostavnim, mi ćemo upotrebiti intent objekat za pokretanje jedne od Androidovih ugrađenih aplikacija, umesto kreiranja nove aplikacije. Za to je potrebna najosnovnija aplikacija, pa pokrenimo novi Android projekat pomoću Android Studioa i dodelimo mu naziv **ActivityStarter**.

Kako da uradite...

Da bismo primer zadržali jednostavnim i mogli da se fokusiramo na zadatak, kreiraćemo funkciju za prikaz intenta u akciji i pozvaćemo ovu funkciju iz dugmeta u aktivnosti.

Kada je novi projekat kreiran u Android Studiou, pratite sledeće korake:

1. Otvorite `MainActivity.java` klasu i dodajte sledeću funkciju:

```
public void launchIntent(View view) {
    Intent intent = new Intent(Intent.ACTION_VIEW);
    intent.setData(Uri.parse("https://www.packtpub.com/"));
    startActivity(intent);
}
```

   Dok kucate ovaj kod, Android Studio će prikazati upozorenje u prikazu i intentu: „Cannot resolve symbol 'intent'".

   To znači da treba da dodate referencu biblioteke u ovaj projekat. Možete to da uradite ručno, tako što ćete u odeljak `import` uneti sledeći kod:

```
import android.view.View;

import android.content.Intent;
```

   Alternativno, samo kliknite na reči napisane crvenim slovima, pritisnite **Alt + Enter** i Android Studio će dodati referencu biblioteke, umesto vas.

2. Otvorite `activity_main.xml` fajl i dodajte sledeći XML:

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Launch Browser"
```

```
android:id="@+id/button"
android:layout_centerVertical="true"
android:layout_centerHorizontal="true"
android:onClick="launchIntent"/>
```

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
    android:layout_height="match_parent" android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    android:paddingBottom="16dp" tools:context=".MainActivity">

    <TextView android:text="Hello world!" android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Launch Browser"
        android:id="@+id/button"
        android:layout_centerVertical="true"
        android:layout_centerHorizontal="true"
        android:onClick="launchIntent"/>

</RelativeLayout>
```

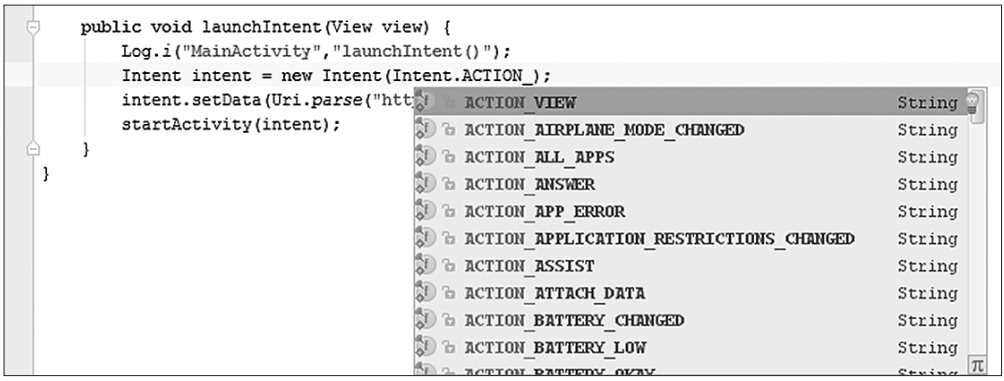
3. Sada je vreme da pokrenete aplikaciju i vidite intent u akciji. Treba ili da kreirate Android emulator (u Android Studiou kliknite na **Tools | Android | AVD Manager**) ili da povežete fizički uređaj na računar.
4. Kada pritisnete dugme **Launch Browser**, otvoriće se standardni web pretraživač sa specifikovanim URL-om.

Kako funkcioniše...

Iako jednostavna, ova aplikacija prikazuje veći deo moći Android operativnog sistema. `Intent` objekat je samo objekat poruke. `Intent` objekti mogu da se upotrebe za komunikaciju između komponenata aplikacije (kao što su servisi i prijemnici za emitovanje) i za komunikaciju između drugih aplikacija na uređaju (kao što smo uradili u ovom receptu).

Za testiranje na fizičkom uređaju možda će biti potrebno da instalirate drajvere za uređaj (drajveri su specifični za proizvođača hardvera). Takođe treba da uključite Developer Mode na uređaju. Uključivanje Developer Modea varira, u zavisnosti od verzije Android operativnog sistema. Ako ne vidite opciju Developer Mode u podešavanjima uređaja, otvorite opciju About Phone i počnite da kucate Build Number. Nakon tri otkucaja, trebalo bi da vidite Toast poruku da ste na dobrom putu da postanete programer. Još četiri otkucaja i opcija će biti uključena.

U ovom receptu kreirali smo intent objekat, tako što smo specifikovali `ACTION_VIEW` kao ono što želimo da uradimo (naša namera) Možda ste primetili kada ste ukucali `Intent`, a zatim tačku, da Android Studio prikazuje padajuću listu mogućnosti (ovo je funkcija automatskog završavanja), kao što je ova:



`ACTION_VIEW`, zajedno sa URL-om u podacima, ukazuje da je namera prikaz web sajta, pa je, stoga, pokrenut standardni pretraživač (drugačiji podaci mogu da pokrenu druge aplikacije). U ovom primeru naša namera je samo da prikažemo URL, pa možemo da pozovemo nameru pomoću metoda `startActivity()`. Postoje i drugi načini za pozivanje namere, u zavisnosti od potreba. U receptu *Vraćanje rezultata iz aktivnosti* upotrebićemo metod `startActivityForResult()`.

Postoji i više...

Uobičajeno je za Android korisnike da preuzimaju omiljene aplikacije za web pretraživanje, snimanje fotografija, slanje tekstualnih poruka i tako dalje. Korišćenjem intenta (namere) možete da obezbedite da aplikacija upotrebi omiljene aplikacije korisnika, umesto da pokušate da ponovo osmislite ovu funkciju.

Takođe vidite

Da biste pokrenuli aktivnost iz selekcije menija, pogledajte recept „*Rukovanje selekcijama menija*“ u Poglavlju 4, „*Meniji*“.

PREKOPČAVANJE IZMEĐU AKTIVNOSTI

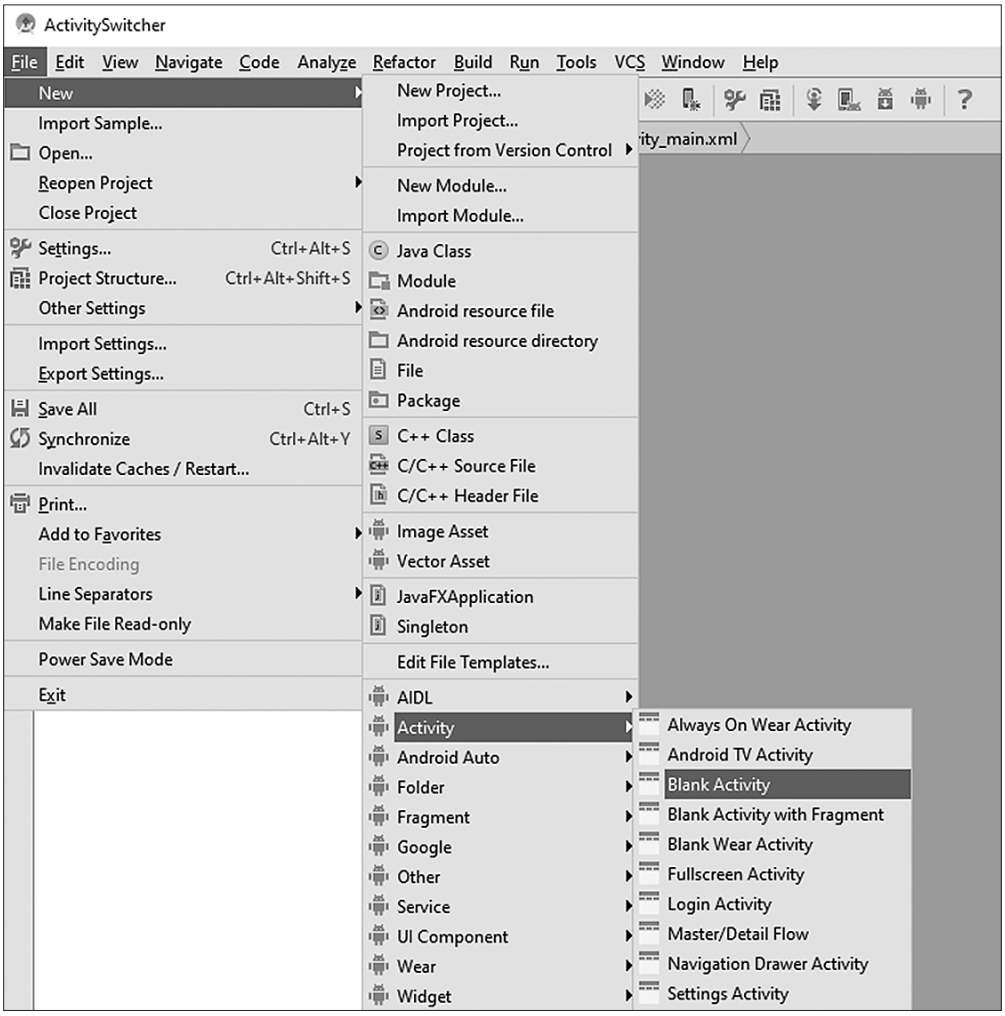
Često ćemo želeti da aktiviramo jednu aktivnost iz druge. Iako to nije težak zadatak, potrebno je izvršiti malo više podešavanja nego što je urađeno u prethodnom receptu, pošto su potrebne dve aktivnosti. Mi ćemo kreirati dve klase aktivnosti i deklarisaćemo obe u manifestu. Takođe ćemo kreirati dugme, kao što smo uradili u prethodnom receptu, za menjanje aktivnosti.

Priprema

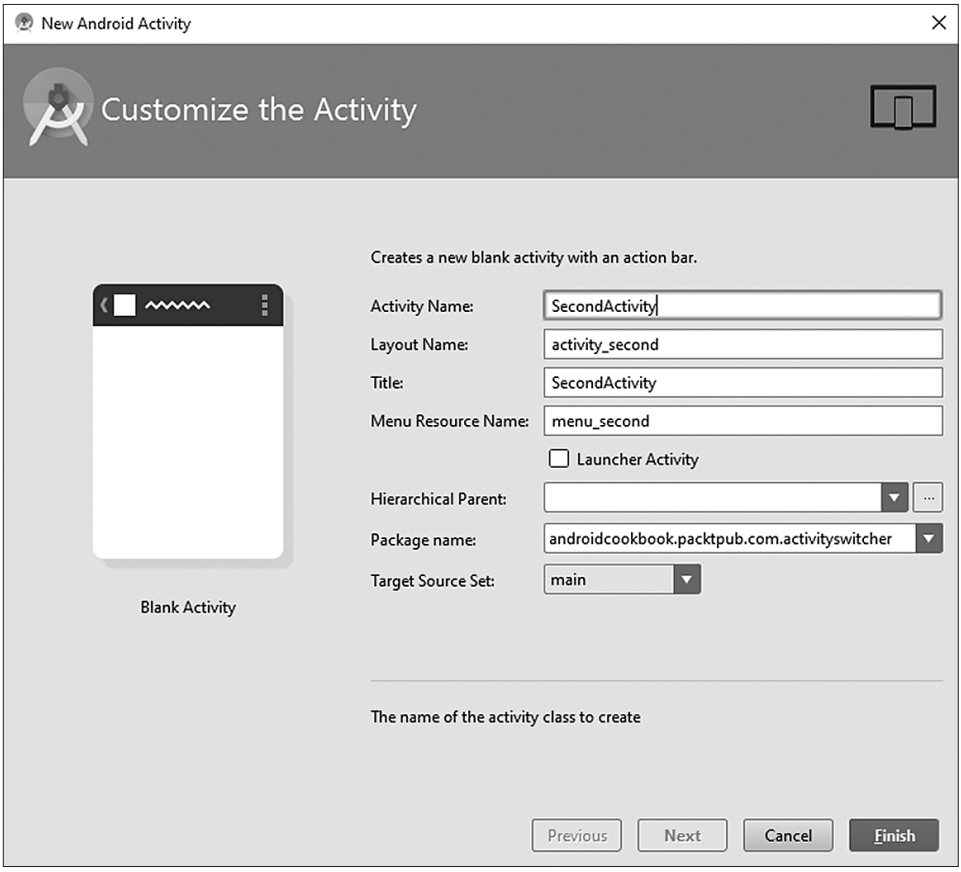
U Android Studiou ćemo kreirati novi projekat, kao što smo uradili u prethodnom receptu, i nazvaćemo ga `ActivitySwitcher`. Android Studio će kreirati prvu aktivnost `ActivityMain` i automatski će je deklarisati u manifestu.

Kako da uradite...

1. Pošto je Android Studio New Project wizard već kreirao prvu aktivnost, treba samo da kreirate drugu. Otvorite projekat **ActivitySwitcher** i kliknite na **File | New | Activity | Blank Activity**, kao što je prikazano na sledećoj slici:



2. U okviru za dijalog **Customize the Activity** možete da ostavite standardno podešavanje `Activity` **Name** kako jeste, a to je **Main2Activity**, ili da promenite naziv na **SecondActivity**, kao što je prikazano na sledećoj slici:



3. Otvorite fajl `MainActivity.java` i dodajte sledeću funkciju:

```
public void onClickSwitchActivity(View view) {
    Intent intent = new Intent(this, SecondActivity.class);
    startActivity(intent);
}
```

4. Sada otvorite fajl `activity_main.xml`, koji se nalazi u direktorijumu `\res\layout`, a da biste kreirali dugme, dodajte sledeći XML:

```
<Button
    android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
```

```
    android:layout_centerVertical="true"
    android:layout_centerHorizontal="true"
    android:text="Launch SecondActivity"
    android:onClick="onClickSwitchActivity"/>
```

5. Možete da pokrenete kod - videćete da će se pojaviti nova aktivnost. Sada nastavite malo dalje i dodajte dugme za zatvaranje `SecondActivity`, što će vas vratiti nazad na prvu aktivnost. Otvorite fajl `SecondActivity.java` i dodajte sledeću funkciju:

```
public void onClickClose(View view) {
    finish();
}
```

6. Na kraju, dodajte dugme **Close** u raspored `SecondActivity`. Otvorite fajl `activity_second.xml` i dodajte sledeći `<Button>` element odmah iza elementa `<TextView>` koji je automatski generisan:

```
<Button
    android:id="@+id/buttonClose"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Close"
    android:layout_centerVertical="true"
    android:layout_centerHorizontal="true"
    android:onClick="onClickClose"/>
```

7. Pokrenite aplikaciju na uređaju ili na emulatoru i vidite dugme u akciji.

Kako funkcioniše…

Stvarno izvršenje zadatka ove vežbe se nalazi u metodu `onClickSwitchActivity()` iz koraka 3. To je mesto gde ćemo deklarisati drugu aktivnost za nameru, koristeći klasu `SecondActivity.class`. Otišli smo i korak dalje, tako što smo dodali dugme za zatvaranje u drugu aktivnost da bismo prikazali uobičajenu situaciju iz stvarnog sveta – pokretanje nove aktivnosti, zatim zatvaranje te aktivnosti i vraćanje originalnoj aktivnosti. Ovo ponašanje je postignuto u funkciji `onClickClose()`. Sve što ova funkcija radi je pozivanje funkcije `finish()`, koja govori sistemu da smo završili konkretnu aktivnost. Finish nas, u stvari, ne vraća na pozivajuću aktivnost ili bilo koju drugu aktivnost; samo zatvara aktuelnu aktivnost i oslanja se na prethodnu. Ako želimo specifičnu aktivnost, možemo ponovo da upotrebimo intent objekat (mi smo samo promenili naziv klase u toku kreiranja namere).

Ovo menjanje aktivnosti neće kreirati uzbudljivu aplikaciju. Aktivnost ne radi ništa osim što demonstrira kako da menjate prikaz sa jedne aktivnosti na drugu, što, naravno, predstavlja formu osnovnog aspekta skoro svih aplikacija koje kreiramo.

Ako smo ručno kreirali aktivnosti, treba da ih dodamo u manifest. Koristeći ove korake, Android Studio se već pobrinuo za XML. Da biste videli šta je Android Studio uradio, otvorite fajl `AndroidManifest.xml` i pogledajte element `<application>`:

```
<activity
    android:name=".MainActivity"
    android:label="@string/app_name">
    <intent-filter>
        <action android:name="android.intent.action.MAIN"/>
        <category android:name="android.intent.category.LAUNCHER/>
    </intent-filter>
</activity>
<activity
    android:name=".SecondActivity"
    android:label="@string/title_activity_second">
</activity>
```

Ono što treba da zapamtite u prethodnom, automatski generisanom kodu je činjenica da druga aktivnost nema `<intent-filter>` element. Glavna aktivnost je generalno ulazna tačka kada pokrećete aplikaciju. Zbog toga su `MAIN` i `LAUNCHER` definisani – da bi sistem znao koju aktivnost da pokrene kada se otvori aplikacija.

Takođe vidite

Da biste naučili više o ugrađivanju vidžeta kao što je `Button`, pogledajte Poglavlje 3, „*Prikazi, vidžeti i stilovi*“.

PROSLEĐIVANJE PODATAKA DRUGOJ AKTIVNOSTI

`Intent` objekat je definisan kao objekat poruke - njegova namena je da komunicira sa drugim komponentama aplikacije. U ovom receptu prikazaćemo kako da prosledite informacije pomoću intent objekta i kako da ih ponovo vratite.

Priprema

Ovaj recept će biti nastavak prethodnog. Nazvaćemo ovaj projekat SendData.

Kako da uradite…

Pošto je ovaj recept nadgrađen na prethodni, veći deo posla je već obavljen. Dodaćemo element `EditText` u glavnu aktivnost, tako da imamo nešto da pošaljemo elementu `SecondActivity`. Upotrebićemo (automatski generisan) `TextView` prikaz za prikaz poruke. Evo i koraka koje treba da pratite:

1. Otvorite fajl `activity_main.xml`, uklonite postojeći element `<TextView>` i dodajte sledeći element `<EditText>`:

```
<EditText
    android:id="@+id/editTextData"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"/>
```

2. Sada otvorite fajl `MainActivity.java` i promenite metod `onClickSwitchActivity()` na sledeći način:

```
public void onClickSwitchActivity(View view) {
    EditText editText = (EditText)findViewById(R.
id.editTextData);
    String text = editText.getText().toString();
    Intent intent = new Intent(this, SecondActivity.class);
    intent.putExtra(Intent.EXTRA_TEXT,text);
    startActivity(intent);
}
```

3. Zatim, otvorite fajl `activity_second.xml` i modifikujte element `<TextView>`, tako da uključuje ID atribut:

```
<TextView
    android:id="@+id/textViewText"
    android:text="@string/hello_world"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>
```

4. Poslednja promena je da modifikujete drugu aktivnost da traži ove nove podatke i da ih prikaže na ekranu. Otvorite fajl `SecondActivity.java` i editujte funkciju `onCreate()` na sledeći način:

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_second);
    TextView textView = (TextView)findViewById(
        R.id.textViewText);
    if (getIntent()!=null && getIntent().hasExtra(
        Intent.EXTRA_TEXT)) {
        textView.setText(getIntent().getStringExtra(
            Intent.EXTRA_TEXT));
    }
}
```

5. Sada pokrenite projekat. Ukucajte tekst u glavnoj aktivnosti i pritisnite **Launch Second Activity** da biste videli da ona šalje podatke.

Kako funkcioniše…

Kao što je i očekivano, intent objekat obavlja sav posao. Kreirali smo intent objekat isto kao u prethodnom receptu, a zatim smo dodali neke podatke. Da li ste primetili poziv metoda `putExtra()`? U našem primeru upotrebili smo već definisan `Intent.EXTRA_TEXT` kao identifikator, ali nismo morali. Možemo da upotrebimo bilo koji taster (već ste sigurno videli ovaj koncept ukoliko su vam poznati parovi naziv/vrednost).

Ključna tačka u vezi upotrebe parova naziv/vrednost je što treba da upotrebite isti naziv da biste dobili nazad podatke. Zbog toga smo upotrebili isti identifikator ključa kada smo čitali dodatne podatke pomoću funkcije `getStringExtra()`.

Druga aktivnost je pokrenuta pomoću intent objekta koji smo kreirali, pa je, u stvari, potrebno da dobijemo intent objekat i da proverimo podatke koji su pomoću njega poslati, što smo uradili u metodu `onCreate()`:

```
textView.setText(getIntent().getStringExtra(Intent.EXTRA_TEXT));
```

Postoji i više…

Nismo ograničeni na slanje samo `String` podataka. `Intent` objekat je veoma fleksibilan i već podržava sve osnovne vrste podataka. Vratite se u Android Studio i kliknite na metod `putExtra`. Zatim, pritisnite tastere **Ctrl** i **razmaknicu**. Android Studio će prikazati listu u kojoj ćete videti različite vrste podataka koje možete da čuvate.

VRAĆANJE REZULTATA IZ AKTIVNOSTI

Mogućnost pokretanja jedne aktivnosti iz druge je dobra, ali ćemo često biti u situaciji kada treba da znamo kako je pozvana aktivnost izvršila svoj zadatak ili, čak, koja aktivnost je pozvana. Metod `startActivityForResult()` pruža nam rešenje.

Priprema

Vraćanje rezultata iz aktivnosti ne razlikuje se mnogo od načina na koji smo pozvali aktivnost u prethodom receptu. Možete da upotrebite projekat iz prethodnog recepta ili da započnete novi projekat pod nazivom `GettingResults`. Bilo koji način da odaberete, kada imate projekat sa dve aktivnosti i sa kodom koji je potreban za pozivanje druge aktivnosti, spremni ste za početak.

Kako da uradite…

Postoji nekoliko promena koje treba da izvršite da biste dobili rezultate:

1. Pre svega, otvorite fajl `MainActivity.java` i dodajte sledeću konstantu u klasu:

```
public static final String REQUEST_RESULT="REQUEST_RESULT";
```

2. Promenite način na koji je pozvan intent objekat, tako što ćete modifikovati metod `onClickSwitchActivity()` da možete očekivati rezultate:

```
public void onClickSwitchActivity(View view) {
    EditText editText = (EditText)findViewById(
        R.id.editTextData);
    String text = editText.getText().toString();
    Intent intent = new Intent(this, SecondActivity.class);
    intent.putExtra(Intent.EXTRA_TEXT,text);
    startActivityForResult(intent,1);
}
```

3. Zatim, dodajte ovaj novi metod da biste primili rezultate:

```
@Override
protected void onActivityResult(int requestCode, int
    resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (resultCode==RESULT_OK) {
        Toast.makeText(this, Integer.toString(
            data.getIntExtra(REQUEST_RESULT, 0)),
            Toast.LENGTH_LONG).show();
    }
}
```

4. Na kraju, modifikujte metod `onClickClose` u fajlu `SecondActivity.java` da biste podesili vraćenu vrednost na sledeći način:

```
public void onClickClose(View view) {
    Intent returnIntent = new Intent();
    returnIntent.putExtra(MainActivity.REQUEST_RESULT,42);
    setResult(RESULT_OK, returnIntent);
    finish();
}
```

Kako funkcioniše…

Kao što vidite, vraćanje rezultata je relativno jednostavno - pozivamo intent objekat pomoću metoda `startActivityForResult`, tako da objekat zna da želimo rezultat. Podesili smo i metod `onActivityResult()` za rukovanje odgovorom za primanje rezultata. Na kraju, upotrebili smo metod `setResult()` da bismo se uverili da druga aktivnost vraća rezultat, pre zatvaranja aktivnosti. U ovom primeru podešavamo rezultat sa statičnom vrednošću. Samo ćemo prikazati ono što primimo da bismo prikazali koncept.

Dobro bi bilo da proverite kod rezultata da biste se uverili da korisnik nije poništio akciju. To je tehnički ceo broj, ali sistem ga koristi kao Bulovu vrednost. Proverite `RESULT_OK` ili `RESULT_CANCEL` i nastavite u skladu sa tim. U našem primeru druga aktivnost nema dugme za poništavanje, pa zašto bismo se trudili da proveravamo? Šta ako korisnik pritisne dugme za vraćanje? Sistem će podesiti kod rezultata na `RESULT_CANCEL`, a intent objekat na nula, što će izazvati kod da generiše izuzetak.

Upotrebili smo objekat **Toast** - on je odgovarajuća poruka koja može da se upotrebi za obaveštenje korisnika. Takođe funkcioniše kao koristan metod za ispravljanje grešaka, pošto nije potreban specijalni raspored ili prostor na ekranu.

Postoji i više…

Osim rezultirajućeg koda, funkcija `onActivityResults()` uključuje i **Request Code**. Verovatno se pitate odakle sad to. To je jednostavno vrednost celog broja, prosleđena pomoću poziva metoda `startActivityForResult()`, koji ima sledeću formu:

```
startActivityForResult(Intent intent, int requestCode);
```

Mi nismo proverili kod zahteva, zato što znamo da treba da rešimo samo jedan rezultat. Međutim, u većim aplikacijama, koje sadrže više aktivnosti, ova vrednost može da se upotrebi za identifikovanje početka zahteva.

Ako je pozvan metod startActivityForResult `()` korišćenjem negativnog koda zahteva, ponašanje je isto kao da je pozvan metod startActivity `()` – odnosno, neće biti vraćen rezultat.

Takođe vidite

- Da biste naučili više o kreiranju novih klasa aktivnosti, pogledajte recept „*Prekopčavanje između aktivnosti*“.
- Za više informacija o Toastu pogledajte recept „*Kreiranje Toasta*“ u Poglavlju 7, „*Upozorenja i obaveštenja*“.

SNIMANJE STATUSA AKTIVNOSTI

Mobilno okruženje je veoma dinamično – u njemu korisnici menjaju zadatke veoma često, češće nego na desktopu. Kada se koristi generalno manje izvora na mobilnom uređaju, moguće je da aplikacija u nekom trenutku postane oštećena. Osim toga, postoji velika mogućnost da će sistem u potpunosti isključiti aplikaciju kako bi aktuelnom zadatku dodelio više izvora, što je potpuno normalno na mobilnom uređaju.

Korisnik će možda započeti da kuca nešto u aplikaciji, biće prekinut telefonskim pozivom ili će se prebaciti na drugu aplikaciju da bi poslao tekstualnu poruku, a kada se vrati u aplikaciju sistem će je možda potpuno isključiti da bi oslobodio memoriju. Da bismo omogućili korisnicima najbolje iskustvo u upotrebi uređaja, potrebno je da olakšamo korisnicima da se vrate u aplikaciju, tamo gde su je i ostavili. Dobro je što Android operativni sistem to olakšava, obezbeđujući povratni poziv za obaveštavanje o promeni statusa aplikacije.

Jednostavno rotiranje uređaja će izazvati operativni sistem da uništi i ponovo kreira aktivnost. To možda izgleda grubo, ali za to postoji dobar razlog – veoma je uobičajeno imati različite rasporede za portretnu i pejzažnu orijentaciju, što obezbeđuje da aplikacije koriste odgovarajuće izvore.

U ovom receptu videćete kako da rukujete povratnim pozivima metoda `onSaveInstanceState()` i `onRestoreInstanceState()` za snimanje statusa aplikacije. Prikazaćemo vam to kreiranjem promenljive kao brojača i povećaćemo broj uvek kada je pritisnuto dugme Count. Takođe ćemo imati vidžete `EditText` i `TextView` da biste videli njihovo ponašanje.

Priprema

U Android Studiou kreirajte novi projekat pod nazivom `StateSaver`. Potrebna vam je samo jedna aktivnost, pa će stoga biti dovoljna automatski generisana glavna aktivnost. Međutim, biće potrebno nekoliko vidžeta, uključujući `EditText`, `Button` i `TextView`. Njihov raspored (u fajlu `activity_main.xml`) će izgledati ovako:

```
<EditText
    android:id="@+id/editText"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_alignParentStart="true"/>

<Button
    android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerInParent="true"
    android:text="Count"
    android:onClick="onClickCounter"/>

<TextView
    android:id="@+id/textViewCounter"
```

```
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_below="@id/button"/>
```

Kako da uradite…

Pratite sledeći set koraka:

1. Da biste pratili brojač, potrebno je da dodate globalnu promenljivu u projekat, zajedno sa ključem za snimanje i vraćanje. U klasu `MainActivity.java` dodajte sledeći kod:

```
static final String KEY_COUNTER = "COUNTER";
private int mCounter=0;
```

2. Zatim, dodajte kod koji je potreban za rukovanje pritiskom na dugme; brojač će se povećavati i biće prikazan rezultat u vidžetu `TextView`:

```
public void onClickCounter(View view) {
    mCounter++;
    ((TextView)findViewById(R.id.textViewCounter)).setText(
    "Counter: " + Integer.toString(mCounter));
}
```

3. Da biste primili obaveštenje o promeni statusa aplikacije, potrebno je da dodate metode `ponSaveInstanceState()` i `onRestoreInstanceState()` u aplikaciju. Otvorite fajl `MainActivity.java` i dodajte sledeće:

```
@Override
protected void onSaveInstanceState(Bundle outState) {
    super.onSaveInstanceState(outState);
    outState.putInt(KEY_COUNTER,mCounter);
}

@Override
protected void onRestoreInstanceState(Bundle
savedInstanceState) {
    super.onRestoreInstanceState(savedInstanceState);
    mCounter=savedInstanceState.getInt(KEY_COUNTER);
}
```

4. Pokrenite program i pokušajte da promenite orijentaciju da biste videli ponašanje (ako koristite emulator, Ctrl + F11 će rotirati uređaj).

Kako funkcioniše…

Svaka aktivnost prolazi kroz više stanja u toku svog „životnog veka“. Podešavanjem povratnih poziva za rukovanje događajima kod može da snimi važne informacije pre nego što je aktivnost poništena.

Korak 3 je, u stvari, mesto na kojem se dešavaju snimanje i vraćanje. Sistem šalje `Bundle` (objekat podataka koji takođe koristi parove naziv/vrednost) u metode. Upotrebite povratni poziv metoda `onSaveInstanceState()` da biste snimili podatke i izvukli ih u povratni poziv metoda `onRestoreInstanceState()`.

Da li ste pokušali da kucate tekst u `EditText` prikaz pre rotiranja uređaja? Ako jeste, onda ste primetili da je tekst takođe vraćen, ali ne postoji kod za rukovanje tim prikazom. Prema standardnom podešavanju, sistem će automatski snimiti stanje - pod uslovom da ima jedinstveni ID (neće svi prikazi automatski snimiti stanje, kao što radi `TextView`, ali stanje možete da snimite ručno ako to želite).

Ako želite da Android automatski snima i vraća stanje prikaza, mora da ima jedinstveni ID (specifikovan pomoću atributa android:id= attribute u rasporedu). Pazite: ne snimaju i ne vraćaju stanje prikaza automatski sve vrste prikaza.

Postoji i više…

Povratni poziv metoda `onRestoreInstanceState()` nije jedino mesto gde stanje može biti vraćeno. Pogledajte potpis metoda `onCreate()`:

```
onCreate(Bundle savedInstanceState)
```

Oba metoda primaju isti primer `Bundle` pod nazivom `savedInstanceState`. Možete da pomerite kod za vraćanje stanja u metod `onCreate()` i on će funkcionisati na isti način. Međutim, paket `savedInstanceState` će biti nula ako ne postoje podaci, kao u toku početka kreiranja aktivnosti. Ako želite da pomerite kod iz povratnog poziva metoda `onRestoreInstanceState()`, samo se uverite da podaci nisu nula, a to ćete uraditi na sledeći način:

```
if (savedInstanceState!=null) {
    mCounter = savedInstanceState.getInt(KEY_COUNTER);
}
```

Takođe vidite

Recept „*Čuvanje podataka trajne aktivnosti*“ će vam predstaviti trajno skladište.

Pogledajte Poglavlje 6, „*Upotreba podataka*“, za više primera o Android aktivnostima.

U receptu „*Razumevanje životnog ciklusa aktivnosti*“ objašnjen je „životni ciklus“ aktivnosti u Androidu.

ČUVANJE PODATAKA TRAJNE AKTIVNOSTI

Mogućnost privremenog čuvanja informacija o aktivnostima je veoma korisna, ali obično ćemo želeti da aplikacija pamti informacije pomoću više sesija.

Android podržava SQLite, ali njegova upotreba može biti preterana za jednostavne podatke, kao što su ime korisnika ili najbolji rezultat. Srećom, Android takođe obezbeđuje jednostavnu opciju za ova scenarija `SharedPreferences`.

Priprema

Možete da upotrebite projekat iz prethodnog recepta ili da započnete novi projekat pod nazivom `PersistentData` (u aplikaciji iz stvarnog sveta verovatno ćete koristiti oba načina). U prethodnom receptu snimili smo `mCounter` u stanje sesije. U ovom receptu dodaćemo novi metod za rukovanje `onPause()` i snimanje stanja `mCounter` u klasu `SharedPreferences`. Vratićemo vrednost u metod `onCreate()`.

Kako da uradite…

Potrebno je da izvršite samo dve promene, i to obe u fajlu `MainActivity.java`:

1. Dodajte sledeći metod `onPause()` da biste snimili podatke pre zatvaranja aktivnosti:

```
@Override
protected void onPause() {
    super.onPause();

    SharedPreferences settings = getPreferences(
        MODE_PRIVATE);
    SharedPreferences.Editor editor = settings.edit();
    editor.putInt(KEY_COUNTER, mCounter);
    editor.commit();
}
```

2. Zatim, dodajte sledeći kod na kraj metoda `onCreate()` da biste vratili brojač:

```
SharedPreferences settings = getPreferences(MODE_PRIVATE);

int defaultCounter = 0;
mCounter = settings.getInt(KEY_COUNTER, defaultCounter);
```

3. Pokrenite program i isprobajte ga.

Kako funkcioniše…

Kao što vidite, ovo je veoma slično snimanju podataka stanja, zato što se takođe koriste parovi naziv/vrednost. Ovde smo sačuvali samo `int`, ali isto tako možemo da snimimo jednu od drugih jednostavnih vrsta podataka. Svaka vrsta podataka ima ekvivalent za dobijanje i podešavanje - na primer, `SharedPreferences.getBoolean()` ili `SharedPreferences.setString()`.

Snimanje podataka zahteva usluge klase `SharedPreferences`.Editor, koja se poziva pomoću metoda `edit()` i prihvata metode `remove()` i `clear()` i metode za podešavanje, kao što je `putInt()`. Imajte na umu da morate da zaključite bilo koje snimanje koje izvršavate pomoću iskaza `commit()`.

Postoji i više…

Postoji i malo sofisticiranija varijanta metoda `getPreferences()`, a to je `getSharedPreferences()`, koji može da se upotrebi za čuvanje više setova parametara.

Upotreba više od jednog fajla parametara

Upotreba metoda `getSharedPreferences()` se ne razlikuje od upotrebe njegovog „suparnika“ `getPreferences()`, ali omogućava da se upotrebi više od jednog fajla parametara; metod ima sledeću formu:

```
getSharedPreferences(String name, int mode)
```

Ovde je `name` fajl. Mode može da bude `MODE_PRIVATE`, `MODE_WORLD_READABLE` ili `MODE_WORLD_WRITABLE`; ovaj parametar opisuje nivo pristupa fajlu.

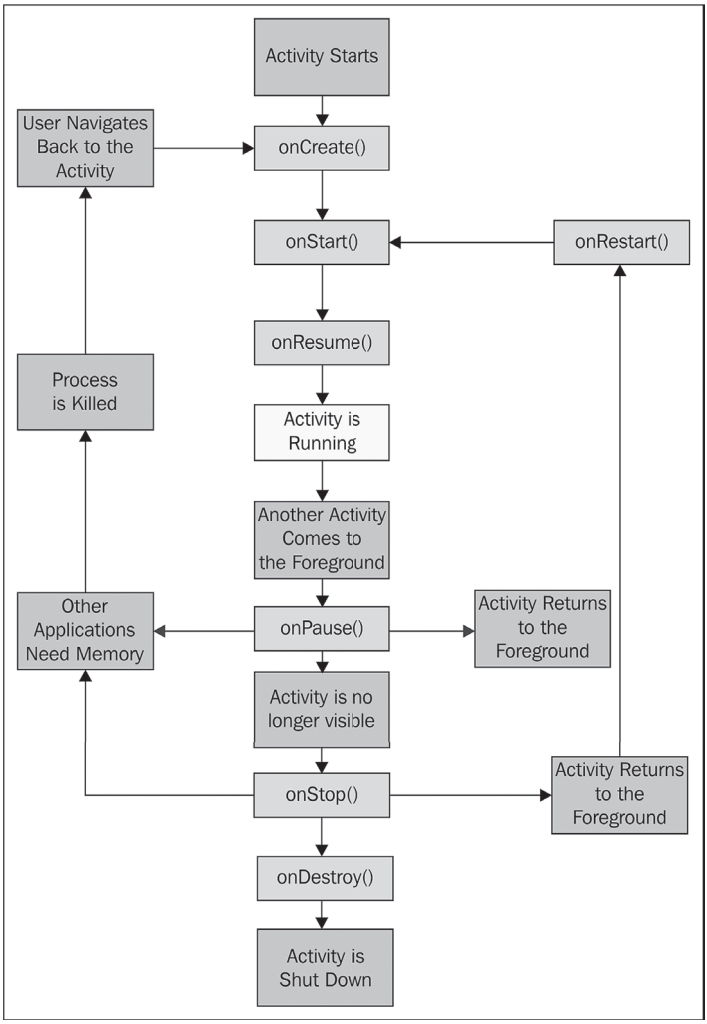
Takođe vidite

Više primera o skladištenju podataka naći ćete u Poglavlju 6, „*Upotreba podataka*“.

RAZUMEVANJE „ŽIVOTNOG CIKLUSA“ AKTIVNOSTI

Android operativni sistem je opasno mesto za aktivnost. On prilično nemilosrdno rukuje zahtevom za resurse na platformi koja se puni baterijom. Aktivnosti mogu da budu izbačene iz memorije kada je memorija prepunjena, bez ikakvog upozorenja i zajedno sa podacima koje aktivnost sadrži. Stoga je važno da razumete „životni ciklus“ aktivnosti.

Na sledećem dijagramu prikazane su faze kroz koje svaka aktivnost prolazi u toku svog „životnog ciklusa“:



Zajedno sa fazama, dijagram takođe prikazuje metode koji mogu da budu poništeni. Kao što možete da vidite, već smo upotrebili većinu ovih metoda u prethodnim receptima.

Priprema

U Android Studiou kreirajte novi projekat, koristeći **Blank Activity**, i dodelite mu naziv `ActivityLifecycle`. Mi ćemo upotrebiti (automatski generisan) `TextView` metod za prikaz informacija o stanju.

Kako da uradite…

Da biste videli kako se aplikacija pomera kroz različite faze, kreirajte metode za sve faze:

1. Otvorite fajl `activity_main.xml` i dodajte ID za automatski generisan `TextView`:

```
android:id="@+id/textViewState"
```

2. Preostali koraci će se izvršavati u fajlu `MainActivity.java`. Dodajte sledeću globalnu deklaraciju:

```
private TextView mTextViewState;
```

3. Modifikujte metod `onCreate()` da biste snimili `TextView` i podesite početni tekst:

```
mTextViewState.setText("onCreate()\n");
```

4. Dodajte sledeće metode za rukovanje ostalim događajima:

```
@Override
protected void onStart() {
    super.onStart();
    mTextViewState.append("onStart()\n");
}

@Override
protected void onResume() {
    super.onResume();
    mTextViewState.append("onResume()\n");
}

@Override
protected void onPause() {
    super.onPause();
    mTextViewState.append("onPause()\n");
}

@Override
```

```
protected void onStop() {
    super.onStop();
    mTextViewState.append("onStop()\n");
}

@Override
protected void onRestart() {
    super.onRestart();
    mTextViewState.append("onRestart()\n");
}

@Override
protected void onDestroy() {
    super.onDestroy();
    mTextViewState.append("onDestroy()\n");
}
```

5. Pokrenite aplikaciju i pratite šta se dešava kada je aktivnost prekinuta pritiskom na dugme Back i Home. Isprobajte i druge aktivnosti, kao što je zadatak prekopčavanja, da biste videli kako to utiče na aplikaciju.

Kako funkcioniše…

Naša aktivnost može da bude aktivna, pauzirana ili zaustavljena. Postoji i četvrto stanje - uništena, ali možemo slobodno da ga ignorišemo:

- Aktivnost se nalazi u stanju active kada je interfejs dostupan za korisnika. Ovo stanje traje od metoda `onResume()` do metoda `onPause()` koji se javlja kada druga aktivnost prelazi u prednji plan. Ako ova nova aktivnost ne remeti u potpunosti našu aktivnost, onda će aktivnost ostati u stanju `paused` dok se nova aktivnost ne završi ili poništi. Tada se odmah poziva metod `onResume()` i aktivnost nastavlja svoj zadatak.
- Kada nova pokrenuta aktivnost popuni ekran ili učini našu aktivnost nevidljivom, onda aktivnost ulazi u fazu `stopped` i njen nastavak će uvek uključivati poziv metoda `onRestart()`.
- Ako se aktivnost nalazi u stanju `paused` ili `stopped`, operativni sistem će je ukloniti iz memorije kada preostane malo prostora u memoriji ili kada je memorija potrebna za druge aplikacije.
- Važno je napomenuti da mi, u stvari, nikada ne vidimo rezultate metoda `onDestroy()`, pošto je aktivnost uklonjena u ovoj tački. Ako želite dalje da istražite ove metode, upotrebite metod `Activity.isFinishing()` da biste videli da li je aktivnost, u stvari, završena pre nego što je izvršen metod `onDestroy()`, kao što možete da vidite u sledećem isečku koda:

```
@Override
  public void onPause() {
  super.onPause();
  mTextView.append("onPause()\n ");
  if (isFinishing()){
    mTextView.append(" ... finishing");
  }
}
```

Kada se implementiraju ovi metodi, uvek se poziva super klasa pre izvršenja bilo kog zadatka

Postoji i više…

Isključivanje aktivnosti

Da biste isključili aktivnost direktno, pozovite njen metod `finish()` koji poziva metod `onDestroy()`. Da biste izvršili istu akciju iz podređene aktivnosti, upotrebite metod `finishFromChild(Activitychild)`, gde je child pozivanje podaktivnosti.

Često je korisno znati da li je aktivnost isključena ili je samo pauzirana, a metod `isFinishing(boolean)` vraća vrednost koja ukazuje u kojoj od ove dve faze se nalazi aktivnost.