

Week 4 - oefening 1: Launching Explicit Intents

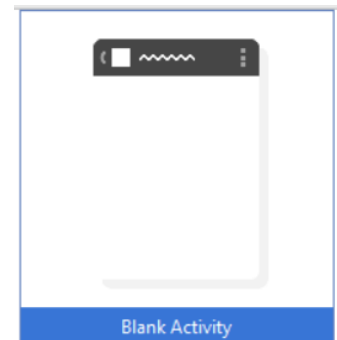
Doelstelling

In dit labo onderzoek je de mogelijkheden om binnen jouw App verschillende activiteiten te beheren, hoe zij kunnen opgeroepen worden, etc door gebruik te maken van explicit intents

Voorbereiding

Maak een nieuw Android Application Project aan met volgende instellingen. Kies voor een projecttype **zonder** fragments

Application Name:	Demo Launching Activities
Package Name:	be.howest.nmct.launching
Target SDK:	21
Activity:	MainActivity – layout: activity_main.xml
Fragments:	
Launch icon:	



Design



De gevraagde (aan te maken) activiteiten vragen geen bijzonder design.

Code

Ga zelf even op onderzoek uit door onderstaande vragen te beantwoorden. De achtergrond (en te kennen leerstof) is te vinden op:

<http://developer.android.com/guide/components/intents-filters.html>

Onderzoek 1: bestudeer het AndroidManifest-file.

- Hoe herken je hierin welke Activity zal opgestart worden? 
- Zijn beide onderdelen binnen de intentfilter wél noodzakelijk? Verwijder één voor één en kijk wat het effect is. 

Onderzoek 2:

- Voeg via Android Studio een tweede (blank) activity 'ExplicitActivity' toe. Je hoeft ook nu niet onmiddellijk een activity met fragment toe te voegen.

Creates a new blank activity with an action bar.

Blank Activity

Activity Name:

Layout Name:

Title:




Menu Resource Name:

☐ Launcher Activity

Hierarchical Parent:

Package name:

Pas het aanwezige textView in deze activity aan zodat je beide activiteiten van elkaar kan onderscheiden.

- Bekijk het android manifest file. Heeft deze activity een Intent filter? 
- Wat gebeurt er als beide activiteiten dezelfde intent filter hebben? 
- Pas de volgorde even aan... wat is het effect? 

Onderzoek 3:

Verwijder de toegevoegde intent-filter van de tweede activity. Plaats op de **eerste** activity een button langswaar de tweede activity via een **explicit intent** opgestart wordt:

```
public class MainActivity extends Activity {

    private Button buttonStartActivity2;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        buttonStartActivity2 = (Button) findViewById(R.id.buttonStartActivity2);
        buttonStartActivity2.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(MainActivity.this, ExplicitActivity.class);
                startActivity(intent);
            }
        });
    }
}
```

Onderzoek 4:

We wensen nu ook data van de ene activity naar de andere activity door te sturen (via explicit intent). Maak hiertoe in de tweede activity een string constante 'EXTRA_INFO' aan.

```
public static final String EXTRA_INFO = "be.howest.nmct.android.launching.EXTRA_INFO";
```

Deze constante gebruiken we als key om een value door te geven. Hieronder vindt u een voorbeeld.

```
Intent intent = new Intent(MainActivity.this, ExplicitActivity.class);
intent.putExtra(ExplicitActivity.EXTRA_INFO, "2NMCT");
startActivity(intent);
```

Vraag nu zelf in de onCreate van de tweede activity deze waarde op via onderstaande code en zorg

dat de waarde weergegeven wordt in de TextView die reeds op je activity stond.

```
String value = getIntent().getStringExtra(ExplicitActivity.EXTRA_INFO);
```

Onderzoek 5:

In sommige situaties wensen we informatie (hier enkel de status ok/cancel) vanuit de tweede activity terug aan de eerste activity te geven. Definieer hiertoe in de eerste activity een int constante 'REQUEST_CODE_EXPLICIT' en geef deze de waarde 1.

```
public static final int REQUEST_CODE_EXPLICIT = 1;
```

Pas de methode 'startSecondActivity' als volgt aan. Wat is het nut van de tweede parameter bij de methode 'startActivityForResult'?

```
public void startSecondActivity(View v){
    Intent intent = new Intent(MainActivity.this, ExplicitActivity.class);
    intent.putExtra(ExplicitActivity.EXTRA_INFO, "2NMCT");
    //startActivity(intent);
    startActivityForResult(intent, REQUEST_CODE_EXPLICIT);
}
```

Plaats in de layout van ExplicitActivity twee buttons 'OK' en 'Cancel' die elk hun eigen methode oproepen.

```
btnOk = (Button) findViewById(R.id.btnOk);
btnOk.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        setResult(RESULT_CANCELED);
        finish();
    }
});

btnCancel = (Button) findViewById(R.id.btnCancel);
btnCancel.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        setResult(RESULT_OK);
        finish();
    }
});
```

Meer toelichting voor alle methodes vind je op:

<http://developer.android.com/reference/android/app/Activity.html>

Het resultaat opvangen in de eerste activity gebeurt als volgt:

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    switch (requestCode) {
        //controlleren van welke vraag we antwoord hebben binnen gekregen
        case REQUEST_CODE_EXPLICIT:
            //wat was het antwoord?
            switch (resultCode) {
                case RESULT_CANCELED:
                    Toast.makeText(MainActivity.this, "User selects OK", Toast.LENGTH_SHORT).show();
                    break;
                case RESULT_OK:
                    Toast.makeText(MainActivity.this, "User selects Cancel", Toast.LENGTH_SHORT).show();
                    break;
            }
            break;
        default:
            super.onActivityResult(requestCode, resultCode, data);
    }
}
```

Meer toelichting is te vinden op:

<http://developer.android.com/training/basics/intents/result.html>

Onderzoek 6:

Uiteraard kan een tweede activity ook andere antwoord dan een 'standaard-antwoord' (zoals RESULT_OK of RESULT_CANCELED) aan de eerste activity terug bezorgen.

De eigen resultCode maak je bovenaan als een final int constante in de tweede activity aan.

Voeg op de tweede activity een derde button toe dat jouw eigen resultCode nu terug geeft.

Pas vervolgens de methode onActivityResult in de eerste activity aan.

Onderzoek 7:

Wellicht merkte je reeds op dat de methode 'onActivityResult' ook een derde parameter heeft. Gebruik deze om extra info vanuit tweede activity naar eerste terug te bezorgen.

Maak hiervoor een Intent aan. Steek in het intent een string en een getal.

```
btnNoIdea.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        //setResult(RESULT_CODE_NOIDEA);
        Intent intent = new Intent();
        intent.putExtra(MainActivity.EXTRA_INFO_BACK_LASTNAME, "Walcarius");
        intent.putExtra(MainActivity.EXTRA_INFO_BACK_AGE, 40);
        setResult(ExplicitActivity.RESULT_CODE_NOIDEA, intent);
        finish();
    }
});
```

Haal deze in de methode 'onActivityResult' uit de MainActivity via de parameter data opnieuw eruit.

```
case ExplicitActivity.RESULT_CODE_NOIDEA:
    //Toast.makeText(MainActivity.this, "User has no idea", Toast.LENGTH_SHORT).show();
    String value = data.getStringExtra(MainActivity.EXTRA_INFO_BACK_LASTNAME);
    int age = data.getIntExtra(MainActivity.EXTRA_INFO_BACK_AGE, 0);
    Toast.makeText(MainActivity.this, "No Idea what's happening there, " + value + " - " + age, Toast.LENGTH_SHORT).show();
    break;
```

Onderzoek 8: (uitbreiding)

Werk een klein dialoogvenster uit langs waar de gebruiker een score kan opgeven. Toon nadien het resultaat via een toast in de activity.

Meer info over het gebruik van dialogs vind je op:

<http://developer.android.com/guide/topics/ui/dialogs.html>

Een goed (verouderd) voorbeeld vind je op:

<http://www.mkvong.com/android/android-prompt-user-input-dialog-example/>

Enkele screenshots (voorbeelden)

