Fondamentaux d'Android

4. Activités et intentions (intents)

Hatem Aziza - September 2024

Introduction
Une application se compose généralement de plusieurs écrans vaguement liés les uns aux autres. Chaque écran est une **activité**. En règle générale, une activité dans une application est spécifiée comme activité « principale » (MainActivity.java), qui est présentée à l'utilisateur au lancement de l'application. L'activité principale peut alors démarrer d'autres activités pour réaliser différentes actions.

Chaque fois qu'une nouvelle activité démarre, l'activité précédente est arrêtée, mais le système conserve l'activité dans une pile (la « **back stack** »). Lorsqu'une nouvelle activité démarre, cette nouvelle activité est placée sur la pile arrière et attire l'attention de l'utilisateur. La pile arrière suit la logique de base de la pile « dernier entré, premier sorti ». Lorsque l'utilisateur a terminé l'activité en cours et appuie sur le bouton Retour, cette activité est retirée de la pile et détruite, et l'activité précédente reprend.

Une activité est démarrée ou activée avec une intention (**intent**). Une intention est un message asynchrone que vous pouvez utiliser dans votre activité pour demander une action à une autre activité ou à un autre composant d'application. Vous utilisez une intention pour démarrer une activité à partir d'une autre activité et pour transmettre des données entre les activités.

Un Intent peut être **explicite** ou **implicite** :
- Une intention **explicite** est celle dans laquelle vous connaissez la cible de cette intention. Autrement dit, vous connaissez déjà le nom de classe complet de cette activité spécifique.
- Une intention **implicite** est une intention dans laquelle vous n'avez pas le nom du composant cible, mais vous avez une action générale à effectuer.

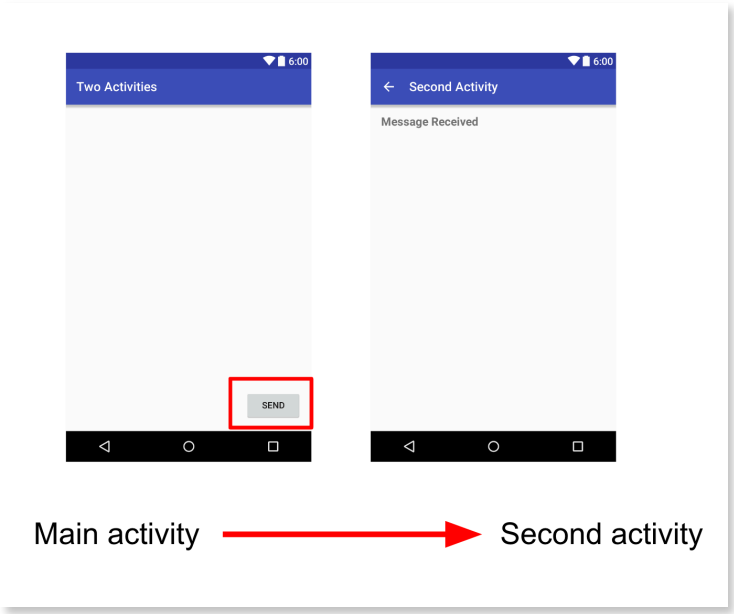
Dans cet atelier, vous créez des intentions explicites.

Ce que vous apprendrez
- Comment créer une nouvelle activité dans Android Studio.
- Comment définir les activités parents et enfants.
- Comment démarrer une activité avec un Intent.
- Comment transmettre des données entre chaque activité avec un Intent explicite.

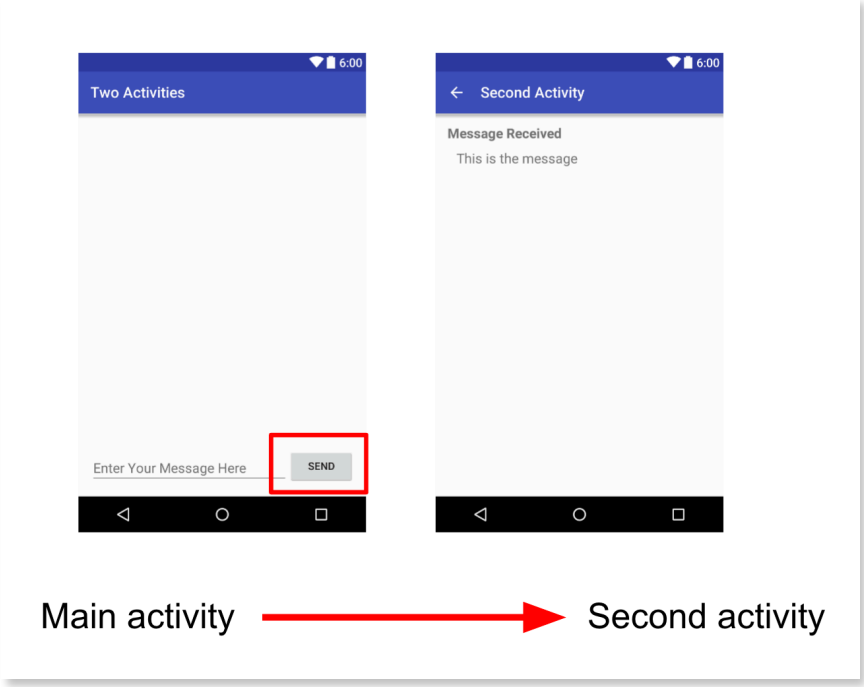
Aperçu de l'application

Dans cet atelier, vous créez et construisez une application appelée **TwoActivities** qui, sans surprise, contient deux implémentations d'activités. Vous créez l'application en trois étapes.

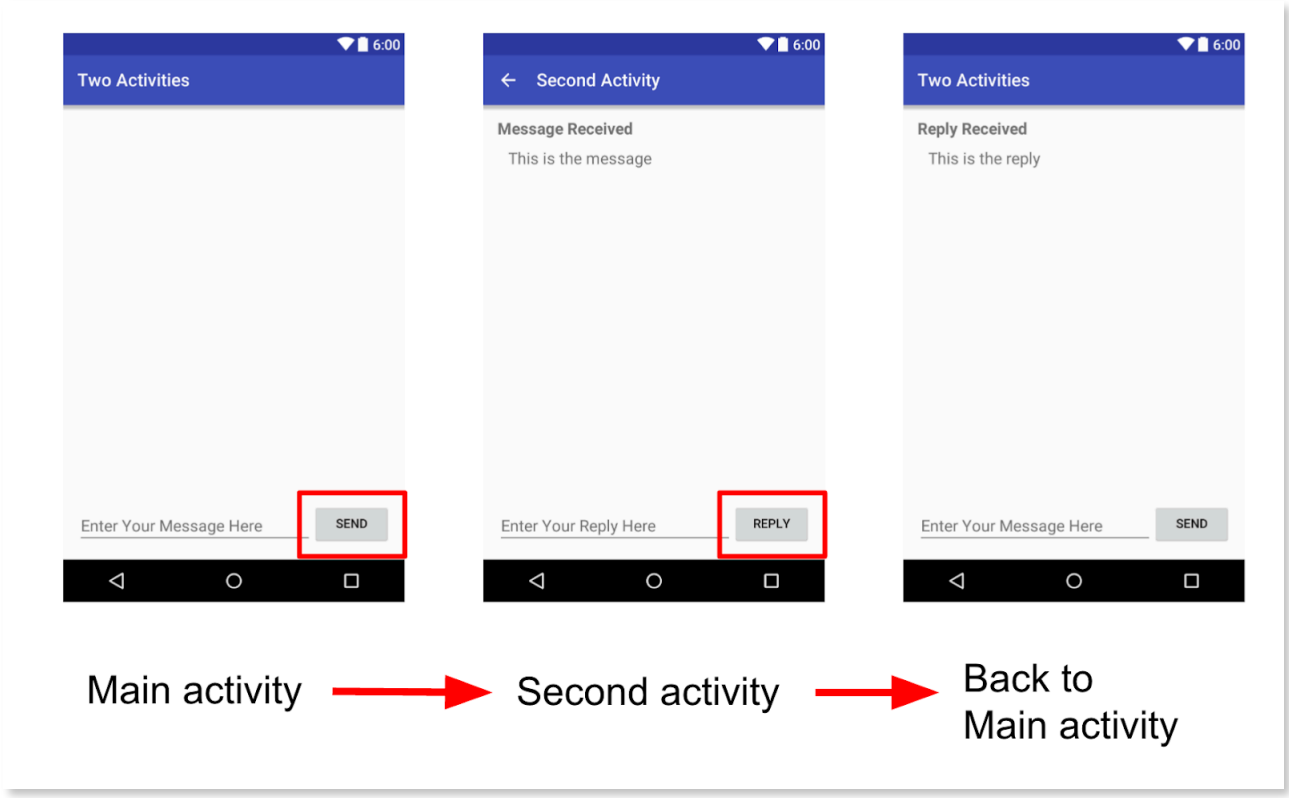
- Dans la première étape, vous créez une application dont l'activité principale contient un bouton, Envoyer. Lorsque l'utilisateur clique sur ce bouton, votre activité principale utilise un Intent pour démarrer la deuxième activité.



- Dans la deuxième étape, vous ajoutez une **EditText** à l'activité principale. L'utilisateur saisit un message et clique sur **SEND**. L'activité principale utilise une intention pour démarrer la deuxième activité et envoyer le message de l'utilisateur à la deuxième activité. La deuxième activité affiche le message qu'elle a reçu.



Dans la dernière étape de la création de l'application, vous ajoutez un **EditText** et un bouton **REPLY** à la deuxième activité. L'utilisateur peut désormais saisir un message de réponse et appuyer sur Répondre , et la réponse est affichée sur l'activité principale. À ce stade, vous utilisez une intention pour transmettre la réponse de la deuxième activité à l'activité principale.



Tâche 1 : Créer le projet TwoActivities

Dans cette tâche, vous configurez le projet initial avec un main Activity, définissez la mise en page et définissez une méthode squelette pour l'événement onClick du bouton.

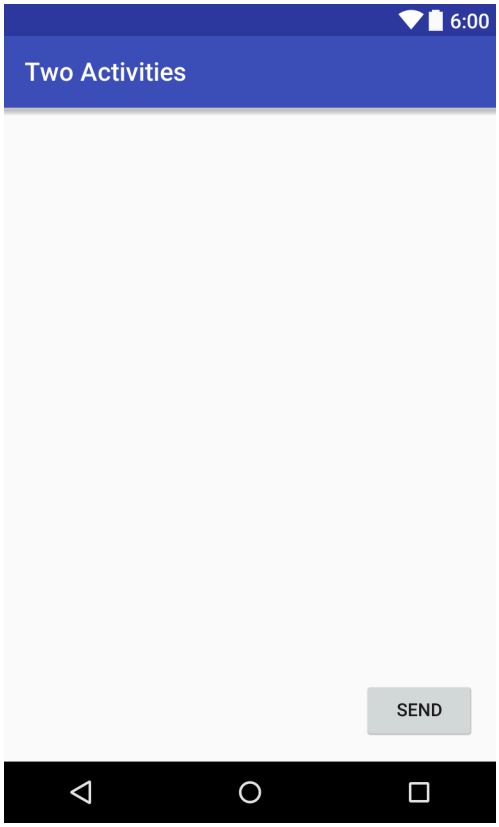
Créer le projet TwoActivities

1. Démarrez Android Studio et créez un nouveau projet Android Studio.
2. Choisissez Activité vide pour le modèle. Cliquez sur Suivant.

Définir le layout de l'activité principale

1. Ouvrez **res > layout >activity_main.xml** dans le volet **Projet > Android**. L'éditeur de mise en page apparaît.
2. Cliquez sur l'onglet **Design** et supprimez le **TextView** (celui qui dit "Hello World").
3. Faites glisser un bouton du volet Palette vers le coin inférieur droit de la mise en page. La connexion automatique crée des contraintes pour le Button.

4. Dans le volet Attributs, définissez **l'ID** sur bouton_main, le **layout_width** et le **layout_height** sur **wrap_content** et saisissez **SEND** pour le champ Texte. La mise en page devrait maintenant ressembler à ceci :



5. Cliquez sur l'onglet Code pour modifier le code XML. Ajoutez l'attribut suivant au Button :

   android:onClick="launchSecondActivity"

   La valeur de l'attribut est soulignée en rouge car la méthode **launchSecondActivity()** n'a pas encore été créée. Ignorez cette erreur pour l'instant ; vous le corrigez dans la tâche suivante.

6. Extrayez la ressource de chaîne (**strings.xml**) pour «**SEND**» et utilisez le nom **button_main** de la ressource.

Le code XML du bouton devrait ressembler à ce qui suit :

```
<Button
    android:id="@+id/button_main"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="16dp"
    android:layout_marginRight="16dp"
    android:text="@string/button_main"
```

```
android:onClick="launchSecondActivity"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintRight_toRightOf="parent" />
```

Définir l'action du bouton

Dans cette tâche, vous implémentez la méthode **launchSecondActivity()** à laquelle vous avez fait référence dans l'attribut **android:onClick**.

1. Cliquez sur "**launchSecondActivity**"dans **activity_main.xml**.
2. Appuyez sur **Alt+Enter** et sélectionnez Créer 'launchSecondActivity(View)' dans 'MainActivity.

Le fichier MainActivity s'ouvre et Android Studio génère une méthode squelette pour launchSecondActivity().

3. À l'intérieur **launchSecondActivity()**, ajoutez une déclaration **Log** indiquant «**Bouton cliqué !** »

   Log.d(LOG_TAG, "Button clicked!");

LOG_TAG s'affichera en rouge. Vous ajouterez la définition de cette variable dans une étape ultérieure.

4. En haut de la classe **MainActivity**, ajoutez une constante pour la variable **LOG_TAG** :

   private static final String LOG_TAG = MainActivity.class.getSimpleName();

Cette constante utilise le nom de la classe elle-même comme balise.

5. Exécutez votre application. Lorsque vous cliquez sur le bouton **SEND** , le message « **Bouton cliqué !** » s'affiche. » dans le volet **Logcat**. S'il y a trop de sortie dans le moniteur, tapez MainActivity dans la zone de recherche et le volet Logcat affichera uniquement les lignes qui correspondent à cette balise.

Le code pour **MainActivity** devrait ressembler à ceci :

```
package com.example.android.twoactivities;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;

public class MainActivity extends AppCompatActivity {
  private static final String LOG_TAG = MainActivity.class.getSimpleName();
```

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

    public void launchSecondActivity(View view) {
        Log.d(LOG_TAG, "Button clicked!");
    }
}
```

Tâche 2 : Créer et lancer la deuxième activité

Dans cette tâche, vous ajoutez une deuxième activité à notre application, avec sa propre mise en page. Vous modifiez le fichier **AndroidManifest.xml** pour définir l'activité principale comme parent de la deuxième activité. Ensuite, vous modifiez la méthode **launchSecondActivity()** pour inclure une intention qui lance la deuxième activité lorsque vous cliquez sur le bouton.

Créez la deuxième activité

1. Cliquez sur le dossier d'application de votre projet et choisissez **Fichier > Nouveau > Activité > Activité vide** .
2. Nommez la nouvelle activité **SecondActivity**. Le nom du layout est renseigné sous la forme **activity_second**.
3. Android Studio ajoute à la fois un nouveau layout (**activity_second.xml**) et un nouveau fichier Java (**SecondActivity.java**) à votre projet pour la nouvelle activité. Il met également à jour le fichier **AndroidManifest.xml** pour inclure la nouvelle activité.

Modifiez le fichier AndroidManifest.XML

1. Ouvrez **AndroidManifest.xml**.
2. Recherchez l'élément **<activity>** créé par Android Studio pour la deuxième activité.

```
<activity android:name=“.SecondActivity"></activity>
```

3. Remplacez l'élément entier <activity> par ce qui suit :

```
<activity android:name=".SecondActivity"
    android:label = "Second Activity"
    android:parentActivityName=".MainActivity">
    <meta-data
        android:name="android.support.PARENT_ACTIVITY"
        android:value="com.example.android.twoactivities.MainActivity" />
</activity>
```

L'attribut **label** ajoute le titre de l'activité à la barre d'application.
Avec l'attribut **parentActivityName** vous indiquez que l'activité principale est le parent de la deuxième activité. Cette relation est utilisée pour la navigation vers le haut dans votre application : la barre d'application de la deuxième activité comportera une flèche orientée vers la gauche afin que l'utilisateur puisse naviguer "vers le haut" jusqu'à l'activité principale.
Avec l'élément **<meta-data>**, vous fournissez des informations arbitraires supplémentaires sur l'activité sous la forme de paires **clé-valeur**. Dans ce cas, les attributs de métadonnées font la même chose que l'attribut **android:parentActivityName**: ils définissent une relation entre deux activités pour une navigation ascendante. Ces attributs de métadonnées sont requis pour les anciennes versions d'Android, car l'attribut **android:parentActivityName** n'est disponible que pour les niveaux d'API 16 et supérieurs.

4. Extrayez une ressource de chaîne pour « **Deuxième activité** » dans le code ci-dessus et utilisez **activity2_name** comme nom de ressource.

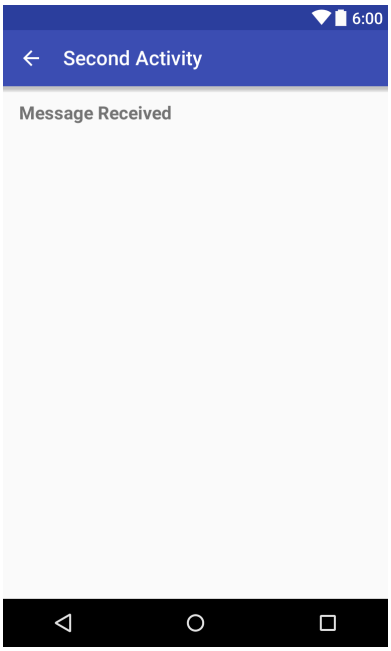
Définir le layout de la deuxième activité

1. Ouvrez **Activity_second.xml** et cliquez sur l'onglet Conception.
2. Faites glisser un **TextView** du volet Palette vers le coin supérieur gauche de la mise en page et ajoutez des contraintes sur les côtés supérieur et gauche de la mise en page. Définissez ses attributs dans le volet Attributs comme suit :

Attribut	Valeur
id	text_header
Marge supérieure	16
Marge de gauche	8
layout_width	wrap_content
layout_height	wrap_content
text	Message Received
textAppearance	AppCompat.Medium
textStyle	B(bold)

La valeur de **textAppearance** est un attribut spécial du thème Android qui définit les styles de police de base.

Le layout devrait maintenant ressembler à ceci :



3. Cliquez sur l'onglet Code pour modifier le code XML et extrayez la chaîne « Message reçu » dans une ressource nommée **text_header**.
4. Ajoutez l'attribut **android:layout_marginLeft="8dp"** au TextView pour compléter l'attribut **layout_marginStart** des anciennes versions d'Android.

Le code XML **activity_second.xml** doit être le suivant :

```
<android.support.constraint.ConstraintLayout xmlns:android="http://
schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.android.twoactivities.SecondActivity">

    <TextView
        android:id="@+id/text_header"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginTop="16dp"
```

```
    android:text="@string/text_header"
    android:textAppearance=
            "@style/TextAppearance.AppCompat.Medium"
    android:textStyle="bold"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
</android.support.constraint.ConstraintLayout>
```

Ajouter un Intent à l'activité principale

Dans cette tâche, vous ajoutez un Intent explicite à l'activité principale. Cet Intent permet d'activer le seconde activité lorsque l'on clique sur le bouton **SEND** .

1. Ouvrez MainActivity .
2. Créez-en un Intent dans la méthode **launchSecondActivity()**.

Le constructeur Intent prend deux arguments pour un Intent explicite: une application Context et le composant spécifique qui la recevra Intent. Ici, vous devez utiliser **this** comme **Context**, et **SecondActivity.class** comme classe spécifique :

```
Intent intent = new Intent(this, SecondActivity.class);
```

3. Appelez la méthode **startActivity()** avec le nouvel Intent comme argument.

```
startActivity(intent);
```

4. Exécutez l'application.

Lorsque vous cliquez sur le bouton **SEND**, MainActivity envoie l'Intent et le système Android lance **SecondActivity**, qui apparaît à l'écran. Pour revenir à MainActivity, cliquez sur le bouton Haut (la flèche gauche dans la barre d'applications) ou sur le bouton Retour en bas de l'écran.

Tâche 3 : Envoyer les données de l'activité principale à la deuxième activité

Votre objet Intent peut transmettre des données à l'activité cible de deux manières :
- dans le champ de données ou
- dans les extras d'intention.

Les données d'intention sont un URI indiquant les données spécifiques sur lesquelles agir. Si les informations que vous souhaitez transmettre à une activité via une intention ne sont pas un URI, ou si vous souhaitez envoyer plusieurs informations, vous pouvez plutôt placer ces informations supplémentaires dans les extras .

Les extras d'intention sont des paires **clé/valeur** dans un fichier **Bundle**. Bundle est une collection de données, stockées sous forme de paires clé/valeur.
Pour transmettre des informations d'une activité à une autre, vous placez des clés et des valeurs dans l'Intent supplémentaire Bundle de l'activité d'envoi, puis vous les récupérez dans l'activité de réception.

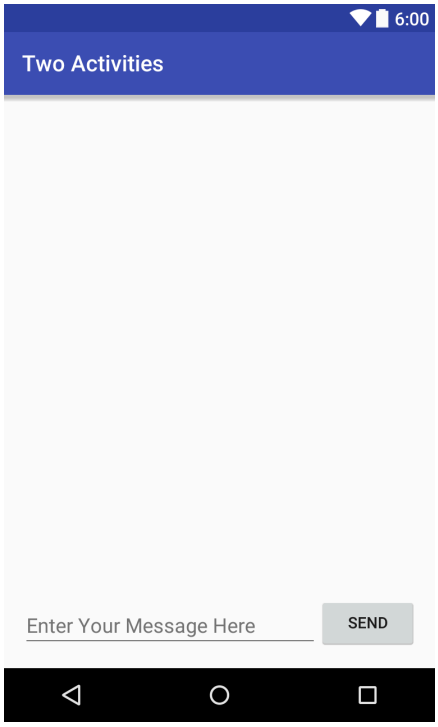
Dans cette tâche, vous modifiez l'Intent explicite pour inclure des données supplémentaires (dans ce cas, une chaîne saisie par l'utilisateur) dans l'Intent extra Bundle. Vous modifiez ensuite SecondActivity pour récupérer ces données hors de l'Intent supplémentaire Bundle et les afficher à l'écran.

Ajoutez un EditText au layout MainActivity

1. Ouvrez Activity_main.xml .
2. Faites glisser un élément Texte brut (**EditText**) du volet Palette vers le bas du layout et ajoutez des contraintes sur le côté gauche du layout, le bas du layout et le côté gauche du bouton **Send**. Définissez ses attributs dans le volet Attributs comme suit :

Attribut	Valeur
id	editText_main
Marge droite	8
Marge de gauche	8
Marge inférieure	16
layout_width	match_constraint
layout_height	wrap_content
inputType	textLongMessage
hint	Enter Your Message Here
text	(Supprimez tout texte dans ce champ)

La nouvelle présentation activity_main.xml ressemble à ceci :



3. Cliquez sur l'onglet Code pour modifier le code XML et extrayez la chaîne «Entrez votre message ici » dans une ressource nommée **editText_main**.

Le code XML du layout devrait ressembler à ce qui suit.

```
<android.support.constraint.ConstraintLayout xmlns:android="http://
schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.android.twoactivities.MainActivity">

    <Button
        android:id="@+id/button_main"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="16dp"
        android:layout_marginRight="16dp"
        android:text="@string/button_main"
        android:onClick="launchSecondActivity"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintRight_toRightOf="parent" />
```

```
    <EditText
        android:id="@+id/editText_main"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginBottom="16dp"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:ems="10"
        android:hint="@string/editText_main"
        android:inputType="textLongMessage"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toStartOf="@+id/button_main"
        app:layout_constraintStart_toStartOf="parent" />
</android.support.constraint.ConstraintLayout>
```

Ajoutez une chaîne aux extras d'Intent

1. Ouvrez MainActivity.
2. Ajoutez une constante public en haut de la classe pour définir la clé de l'extra de l'Intent:

```
public static final String EXTRA_MESSAGE =
        "com.example.android.twoactivities.extra.MESSAGE";
```

3. Ajoutez une variable privée en haut de la classe pour contenir l'EditText :

```
private EditText mMessageEditText;
```

4. Dans la méthode **onCreate()**, utilisez **findViewById()** pour obtenir une référence à l'EditText et assignez-la à cette variable privée :

```
mMessageEditText = findViewById(R.id.editText_main);
```

5. Dans la méthode **launchSecondActivity()**, juste en dessous du new Intent, récupérez le texte de l'EditText sous forme de chaîne :

```
String message = mMessageEditText.getText().toString();
```

6. Ajoutez cette chaîne à l'extra de l'Intent avec la constante EXTRA_MESSAGE comme clé et la chaîne comme valeur :

```
intent.putExtra(EXTRA_MESSAGE, message);
```

La méthode **onCreate()** devrait maintenant ressembler à ceci :

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    mMessageEditText = findViewById(R.id.editText_main);
}
```

La méthode **launchSecondActivity()** devrait maintenant ressembler à ceci :

```
public void launchSecondActivity(View view) {
    Log.d(LOG_TAG, "Button clicked!");
    Intent intent = new Intent(this, SecondActivity.class);
    String message = mMessageEditText.getText().toString();
    intent.putExtra(EXTRA_MESSAGE, message);
    startActivity(intent);
}
```

Ajoutez un TextView à SecondActivity pour le message

1. Ouvrez **Activity_second.xml**.
2. Faites glisser un autre TextView vers le layout située sous le TextView **text_header** et ajoutez des contraintes sur le côté gauche du layout et au bas de text_header.
3. Définissez les nouveaux attributs de TextView dans le volet Attributs comme suit :

Attribut	Valeur
id	text_message
Marge supérieure	8
Marge de gauche	8
layout_width	wrap_content
layout_height	wrap_content
text	(Supprimez tout texte dans ce champ)
textAppearance	AppCompat.Medium

Le nouveau layout est identique à celui de la tâche précédente, car le nouveau TextView ne contient pas (encore) de texte et n'apparaît donc pas à l'écran.
Le code XML de **activity_second.xml** devrait ressembler à ceci :

```
<android.support.constraint.ConstraintLayout xmlns:android="http://
schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.android.twoactivities.SecondActivity">

    <TextView
        android:id="@+id/text_header"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="16dp"
        android:text="@string/text_header"
        android:textAppearance=
                "@style/TextAppearance.AppCompat.Medium"
        android:textStyle="bold"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/text_message"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/text_header" />
</android.support.constraint.ConstraintLayout>
```

Modifiez SecondActivity pour obtenir les extras et afficher le message

1. Ouvrez **SecondActivity** pour ajouter du code à la méthode **onCreate().**
2. Obtenez l'Intent qui a activé cette activité :

```
Intent intent = getIntent();
```

3. Récupérez la chaîne contenant le message des extras Intent en utilisant la variable statique **MainActivity.EXTRA_MESSAGE** comme clé :

    String message = intent.getStringExtra(MainActivity.EXTRA_MESSAGE);

4. Utilisez **findViewByID()** pour obtenir une référence au message TextView à partir du layout :

    TextView textView = findViewById(R.id.text_message);

5. Définissez le texte du TextViewsur la chaîne de l' Intentextra :

    textView.setText(message);

6. Exécutez l'application. Lorsque vous saisissez un message dans MainActivity et cliquez sur **Send**, SecondActivity se lance et affiche le message.

La méthode **onCreate()** de **SecondActivity** devrait ressembler à ceci :

```
@Override
protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_second);
   Intent intent = getIntent();
   String message = intent.getStringExtra(MainActivity.EXTRA_MESSAGE);
   TextView textView = findViewById(R.id.text_message);
   textView.setText(message);
}
```

Tâche 4 : Renvoyer les données à l'activité principale

Maintenant que vous disposez d'une application qui lance une nouvelle activité et lui envoie des données, la dernière étape consiste à renvoyer les données de la deuxième activité à l'activité principale. Vous utilisez également une intention et des extras d'intention pour cette tâche.

Ajoutez un EditText et un Button au layout SecondActivity

1. Ouvrez strings.xml et ajoutez des ressources de chaîne pour le texte du bouton et l'EditText :

    <string name="button_second">Reply</string>
    <string name="editText_second">Enter Your Reply Here</string>
2. Ouvrez **Activity_main.xml** et **Activity_second.xml**.

3. Copiez les éléments EditText et Button du layout **activity_main.xml** et collez- les dans le layout **activity_second.xml**.
4. Dans **activity_second.xml**, modifiez les valeurs d'attribut du bouton comme suit :

Ancienne valeur d'attribut	Nouvelle valeur d'attribut
android:id="@+id/button_main"	android:id="@+id/button_second"
android:onClick="launchSecondActivity"	android:onClick="returnReply"
android:text="@string/button_main"	android:text="@string/button_second"

5. Dans **activity_second.xml**, modifiez les valeurs d'attribut de l'EditText comme suit :

Ancienne valeur d'attribut	Nouvelle valeur d'attribut
android:id="@+id/editText_main"	android:id="@+id/editText_second"
app:layout_constraintEnd_toStartOf ="@+id/button"	app:layout_constraintEnd_toStartOf ="@+id/button_second"
android:hint= "@string/editText_main"	android:hint ="@string/editText_second"

6. Dans l'éditeur de layout XML, cliquez sur **returnReply**, appuyez sur Alt+Enter et sélectionnez Créer '**returnReply(View)**' dans '**SecondActivity**'.

Android Studio génère une méthode **returnReply()**. Vous implémenterez cette méthode dans la tâche suivante.

La nouvelle présentation de **activity_second.xml** ressemble à ceci :

Le code XML de **activity_second.xml** est le suivant :

```
<android.support.constraint.ConstraintLayout xmlns:android="http://
schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.android.twoactivities.SecondActivity">

    <TextView
        android:id="@+id/text_header"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginTop="16dp"
        android:text="@string/text_header"
        android:textAppearance="@style/TextAppearance.AppCompat.Medium"
        android:textStyle="bold"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
```

```
    <TextView
        android:id="@+id/text_message"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginTop="8dp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/text_header" />

    <Button
        android:id="@+id/button_second"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="16dp"
        android:layout_marginRight="16dp"
        android:text="@string/button_second"
        android:onClick="returnReply"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintRight_toRightOf="parent" />

    <EditText
        android:id="@+id/editText_second"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginBottom="16dp"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:ems="10"
        android:hint="@string/editText_second"
        android:inputType="textLongMessage"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toStartOf="@+id/button_second"
        app:layout_constraintStart_toStartOf="parent" />
</android.support.constraint.ConstraintLayout>
```

Créez un Intent de réponse dans la deuxième activité

Les données de réponse de la deuxième activité à la principale sont envoyées en extra Intent. Vous construisez cet Intent de retour et y insérez les données de la même manière que vous le faites pour l'envoi Intent.

1. Ouvrez **SecondActivity**.
2. En haut de la classe, ajoutez une constante publique pour définir la clé de l'extra Intent:

```
public static final String EXTRA_REPLY =
        "com.example.android.twoactivities.extra.REPLY";
```

3. Ajoutez une variable privée en haut de la classe pour contenir l'EditText.

   private EditText mReply;

4. Dans **onCreate()**, avant le code Intent, utilisez **findViewByID()** pour obtenir une référence à EditText et assignez-la à cette variable privée :

   mReply = findViewById(R.id.editText_second);

5. Dans laméthode **returnReply()**, récupérez le texte de l'EditText sous forme de chaîne :

   String reply = mReply.getText().toString();

6. Dans la méthode **returnReply()**, créez une nouvelle intention pour la réponse : ne réutilisez pas l'objet Intent que vous avez reçu de la demande d'origine.

   Intent replyIntent = new Intent();

7. Ajoutez la **replychaîne** de l'EditText à la nouvelle intention en tant que extra de l'Intent. Parce que les extras sont des paires clé/valeur, ici la clé est **EXTRA_REPLY**, et la valeur est la **reply** :

   replyIntent.putExtra(EXTRA_REPLY, reply);

8. Définissez le résultat sur **RESULT_OK** pour indiquer que la réponse a réussi. La classe Activity classe définit les codes de résultat, notamment **RESULT_OK** et **RESULT_CANCELLED**.

   setResult(RESULT_OK,replyIntent);

9. Appelez finish() pour fermer le Activity et revenez au MainActivity.

   finish();

Le code **SecondActivity** devrait maintenant être le suivant :

```
public class SecondActivity extends AppCompatActivity {
    public static final String EXTRA_REPLY =
            "com.example.android.twoactivities.extra.REPLY";
    private EditText mReply;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_second);
        mReply = findViewById(R.id.editText_second);
        Intent intent = getIntent();
        String message = intent.getStringExtra(MainActivity.EXTRA_MESSAGE);
        TextView textView = findViewById(R.id.text_message);
        textView.setText(message);
    }

    public void returnReply(View view) {
        String reply = mReply.getText().toString();
        Intent replyIntent = new Intent();
        replyIntent.putExtra(EXTRA_REPLY, reply);
        setResult(RESULT_OK, replyIntent);
        finish();
    }
}
```

Ajoutez des éléments TextView pour afficher la réponse

MainActivity a besoin d'un moyen d'afficher la réponse envoyée à SecondActivity. Dans cette tâche, vous ajoutez des éléments TextView à activity_main.xml pour afficher la réponse au format MainActivity.
Pour faciliter cette tâche, copiez les éléments TextView que vous avez utilisés dans SecondActivity.

1. Ouvrez strings.xml et ajoutez une ressource de chaîne pour l'en-tête de réponse :

```
<string name="text_header_reply">Reply Received</string>
```

2. Ouvrez Activity_main.xml et Activity_second.xml .
3. Copiez les deux TextView de activity_second.xml et collez-les dans activity_main.xml au-dessus du bouton .
4. Dans activity_main.xml, modifiez les valeurs d'attribut du premier TextView comme suit :

Ancienne valeur d'attribut	Nouvelle valeur d'attribut
android:id="@+id/text_header"	android:id="@+id/text_header_reply"
android:text= "@string/text_header"	android:text= "@string/text_header_reply"

5. Dans activity_main.xml, modifiez les valeurs d'attribut pour le deuxième TextView comme suit:

Ancienne valeur d'attribut	Nouvelle valeur d'attribut
android:id="@+id/text_message"	android:id= "@+id/text_message_reply"
app:layout_constraintTop_toBottom Of="@+id/text_header"	app:layout_constraintTop_toBottomOf=" @+id/text_header_reply"

6. Ajoutez l'attribut android:visibility à chaque TextView pour les rendre initialement invisibles. (Les avoir visibles à l'écran, mais sans aucun contenu, peut prêter à confusion pour l'utilisateur.)

    android:visibility="invisible"

Vous rendrez ces TextView visibles une fois que les données de réponse auront été renvoyées par le second Activity.

activity_main.xml est identique à celle de la tâche précédente, même si vous avez ajouté deux nouveaux TextView au layout. Parce que vous définissez ces éléments comme invisibles, ils n'apparaissent pas à l'écran.

Voici le code XML de activity_main.xml :

```
<android.support.constraint.ConstraintLayout xmlns:android="http://
schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.android.twoactivities.MainActivity">

    <TextView
```

```
    android:id="@+id/text_header_reply"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginLeft="8dp"
    android:layout_marginTop="16dp"
    android:text="@string/text_header_reply"
    android:textAppearance="@style/TextAppearance.AppCompat.Medium"
    android:textStyle="bold"
    android:visibility="invisible"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

<TextView
    android:id="@+id/text_message_reply"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginLeft="8dp"
    android:layout_marginTop="8dp"
    android:visibility="invisible"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/text_header_reply" />

<Button
    android:id="@+id/button2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="16dp"
    android:layout_marginRight="16dp"
    android:text="@string/button_main"
    android:onClick="launchSecondActivity"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintRight_toRightOf="parent" />

<EditText
    android:id="@+id/editText_main"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginBottom="16dp"
    android:layout_marginEnd="8dp"
    android:layout_marginStart="8dp"
    android:ems="10"
    android:hint="@string/editText_main"
```

```
        android:inputType="textLongMessage"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toStartOf="@+id/button2"
        app:layout_constraintStart_toStartOf="parent" />
</android.support.constraint.ConstraintLayout>
```

Récupérez la réponse de l'extra Intent et affichez-la

1. Ouvrez **MainActivity** .
2. Ajoutez une constante publique en haut de la classe pour définir la clé d'un type particulier de réponse qui vous intéresse :

   public static final int TEXT_REQUEST = 1;

3. Ajoutez deux variables privées pour contenir l'en-tête de réponse et les éléments TextView de réponse :

   private TextView mReplyHeadTextView;
   private TextView mReplyTextView;

4. Dans **onCreate()**, utilisez **findViewByID()** pour obtenir des références de la mise en page vers l'en-tête de réponse et les éléments TextView de réponse. Attribuez ces instances de vue aux variables privées :

   mReplyHeadTextView = findViewById(R.id.text_header_reply);
   mReplyTextView = findViewById(R.id.text_message_reply);

La méthode complète **onCreate()** devrait maintenant ressembler à ceci :

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    mMessageEditText = findViewById(R.id.editText_main);
    mReplyHeadTextView = findViewById(R.id.text_header_reply);
    mReplyTextView = findViewById(R.id.text_message_reply);
}
```

5. Dans la méthode **launchSecondActivity()**, remplacez l'appel **startActivity()** par **startActivityForResult()**, et incluez la clé **TEXT_REQUEST** comme argument :

   startActivityForResult(intent, TEXT_REQUEST);

6. Remplacez **onActivityResult()** avec cette signature :

```
@Override
public void onActivityResult(int requestCode,
                    int resultCode, Intent data) {
}
```

Les trois arguments de onActivityResult() contenant toutes les informations dont vous avez besoin pour gérer les données de retour : celui que requestCode vous avez défini lorsque vous avez lancé l'activité avec startActivityForResult(), celui resultCode défini dans le launch Activity(généralement l'un des RESULT_OK ou RESULT_CANCELED) et l'Intent data qui contient les données renvoyées par le launch Activity.

7. À l' intérieur onActivityResult(), appelez super.onActivityResult():

```
super.onActivityResult(requestCode, resultCode, data);
```

8. Ajoutez du code à tester pour TEXT_REQUEST pour vous assurer que vous traitez le bon Intent résultat, au cas où il y en aurait plusieurs. Testez également RESULT_OK, pour vous assurer que la demande a abouti :

```
if (requestCode == TEXT_REQUEST) {
   if (resultCode == RESULT_OK) {
   }
}
```

La classe Activity définit les codes de résultat. Le code peut être RESULT_OK (la demande a réussi), RESULT_CANCELED (l'utilisateur a annulé l'opération) ou RESULT_FIRST_USER (pour définir vos propres codes de résultat).

9. À l'intérieur du bloc if interne, obtenez l'extra Intent de la réponse Intent(data). Ici, la clé de l'extra est la constante **EXTRA_REPLY** de SecondActivity :

```
String reply = data.getStringExtra(SecondActivity.EXTRA_REPLY);
```

10. Définissez la visibilité de l'en-tête de réponse sur true :

```
mReplyHeadTextView.setVisibility(View.VISIBLE);
```

11. Définissez le texte de TextView de la réponse sur reply, et définissez sa visibilité sur true :

```
mReplyTextView.setText(reply);
mReplyTextView.setVisibility(View.VISIBLE);
```

12. La méthode complète onActivityResult() devrait maintenant ressembler à ceci :

```
@Override
public void onActivityResult(int requestCode,
                    int resultCode, Intent data) {
   super.onActivityResult(requestCode, resultCode, data);
   if (requestCode == TEXT_REQUEST) {
      if (resultCode == RESULT_OK) {
         String reply =
               data.getStringExtra(SecondActivity.EXTRA_REPLY);
         mReplyHeadTextView.setVisibility(View.VISIBLE);
         mReplyTextView.setText(reply);
         mReplyTextView.setVisibility(View.VISIBLE);
      }
   }
}
```

13. Exécutez l'application.

Désormais, lorsque vous envoyez un message à la seconde activité et recevez une réponse, la principal activité se met à jour pour afficher la réponse.