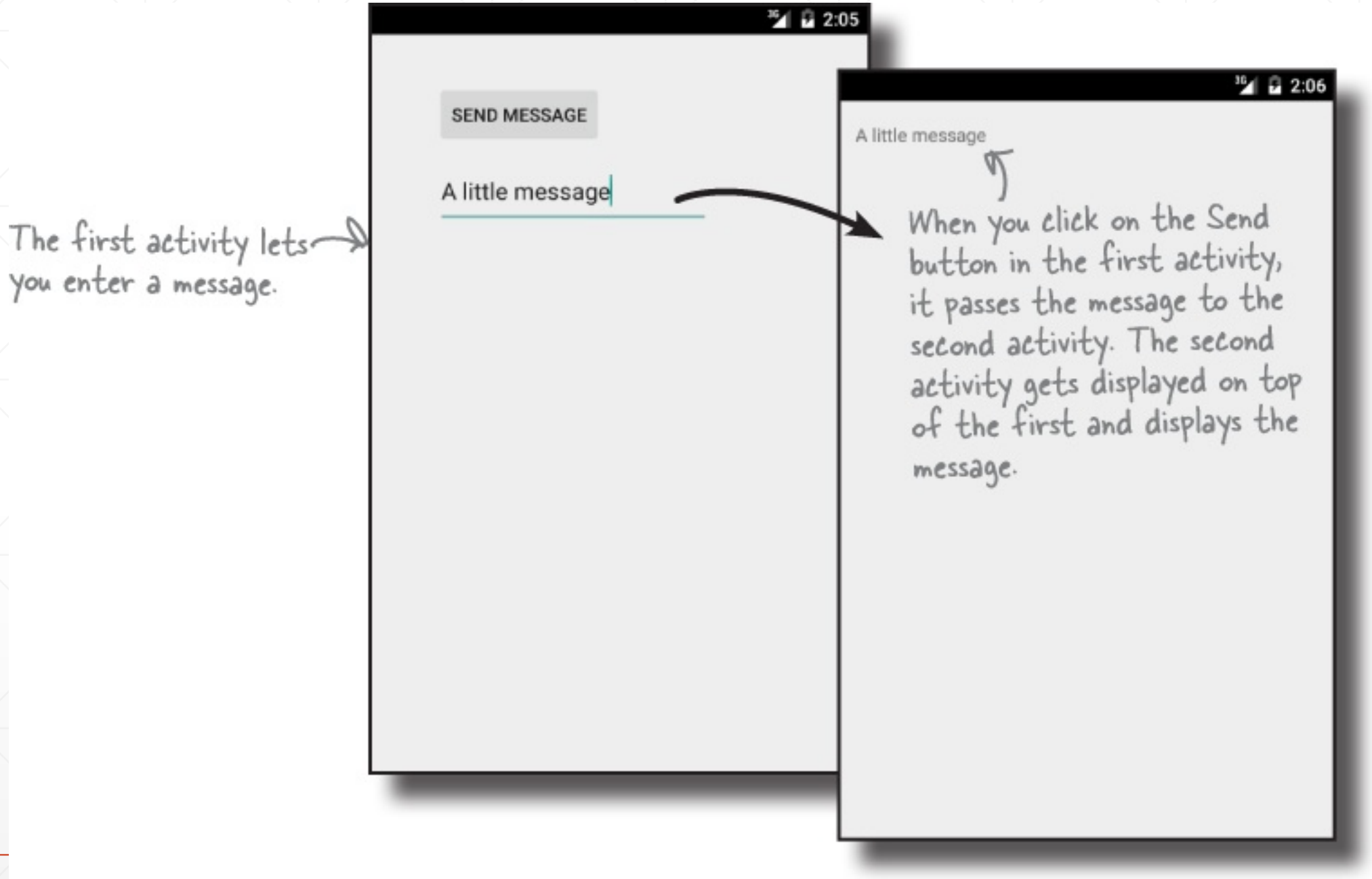
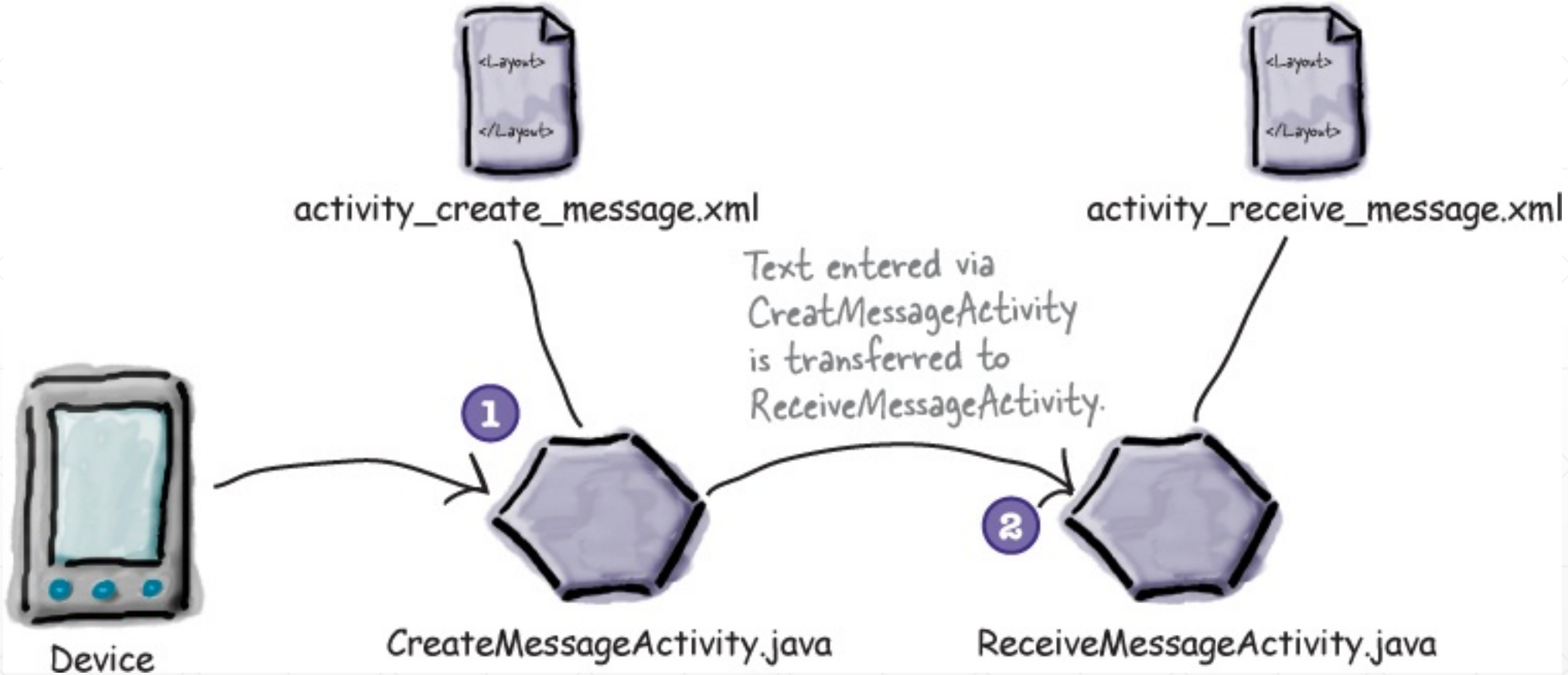

2:05
SEND MESSAGE
A little message
The first activity lets you enter a message.
2:06
A little message
When you click on the Send button in the first activity, it passes the message to the second activity. The second activity gets displayed on top of the first and displays the message.

Multiple Activities



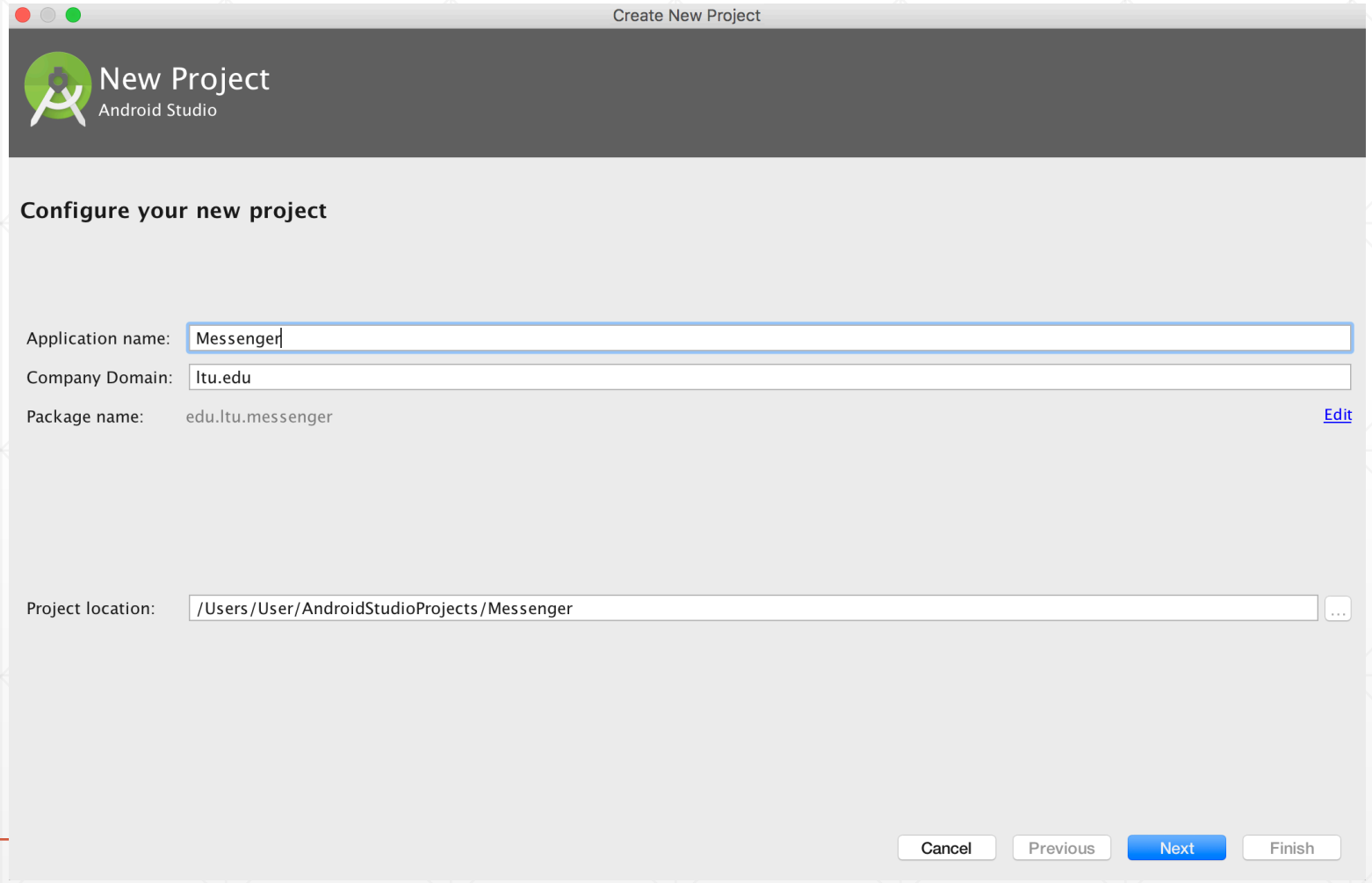
Multiple Activities

- Create 1st activity
- Create 2nd activity
- Call 2nd activity
- Pass Data

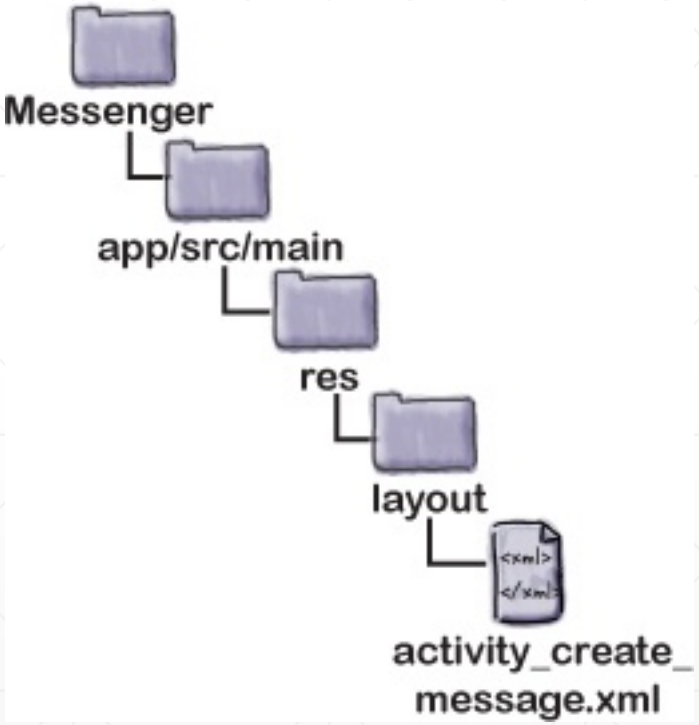
Multiple Activities

- A task is two or more activities chained together.
- The <EditText> element defines an editable text field for entering text. It inherits from the Android View class.
- You can add a new activity in Android Studio by choosing File → New... → Activity.
- Each activity you create must have an entry in AndroidManifest.xml.
- An intent is a type of message that Android components use to communicate with one another.
- An explicit intent explicitly specifies the component the intent is targeted at. You create an explicit intent using Intent intent = new Intent(this, Target.class);
- To start an activity, call startActivity(intent). If no activities are found, it throws an ActivityNotFoundException.
- Use the putExtra() method to add extra information to an intent.
- Use the getIntent() method to retrieve the intent that started the activity

Create 1st Activity



Create 1st Activity




```
<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:paddingBottom="16dp"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    tools:context=".CreateMessageActivity" >

    <Button
        android:id="@+id/send"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        android:layout_marginLeft="36dp"
        android:layout_marginTop="21dp"
        android:onClick="onSendMessage"
        android:text="@string/send" />

    <EditText
        android:id="@+id/message"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/send"
        android:layout_below="@+id/send"
        android:layout_marginTop="18dp"
        android:ems="10" />

</RelativeLayout>
```

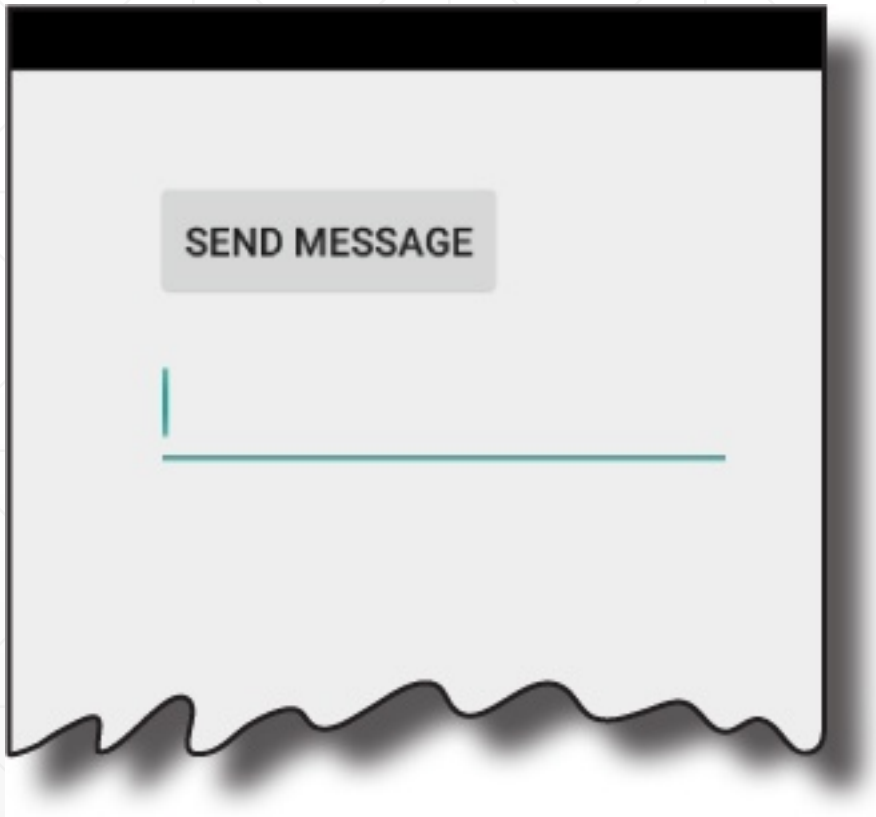
Replace the <TextView> Android Studio gives you with the <Button> and <EditText>.

Clicking on the button runs the onSendMessage() method in the activity.

This is a String resource.

This describes how wide the <EditText> should be. It should be wide enough to accommodate 10 letter M's.

Create 1st Activity




```
<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:paddingBottom="16dp"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    tools:context=".CreateMessageActivity" >

    <Button
        android:id="@+id/send"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        android:layout_marginLeft="36dp"
        android:layout_marginTop="21dp"
        android:onClick="onSendMessage"
        android:text="@string/send" />

    <EditText
        android:id="@+id/message"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/send"
        android:layout_below="@+id/send"
        android:layout_marginTop="18dp"
        android:ems="10" />

</RelativeLayout>
```

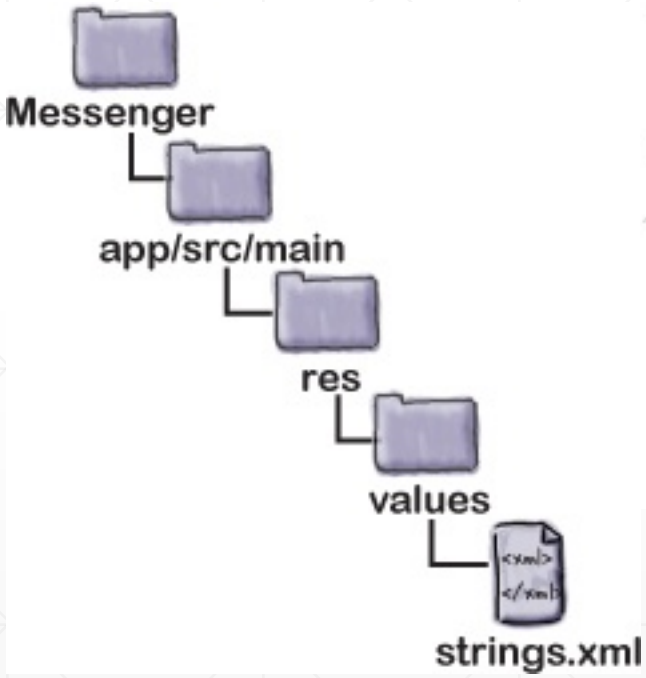
Replace the <TextView> Android Studio gives you with the <Button> and <EditText>.

Clicking on the button runs the onSendMessage() method in the activity.

This is a String resource.

This describes how wide the <EditText> should be. It should be wide enough to accommodate 10 letter M's.

Create 1st Activity




```
...
    <string name="send">Send Message</string>
...
```

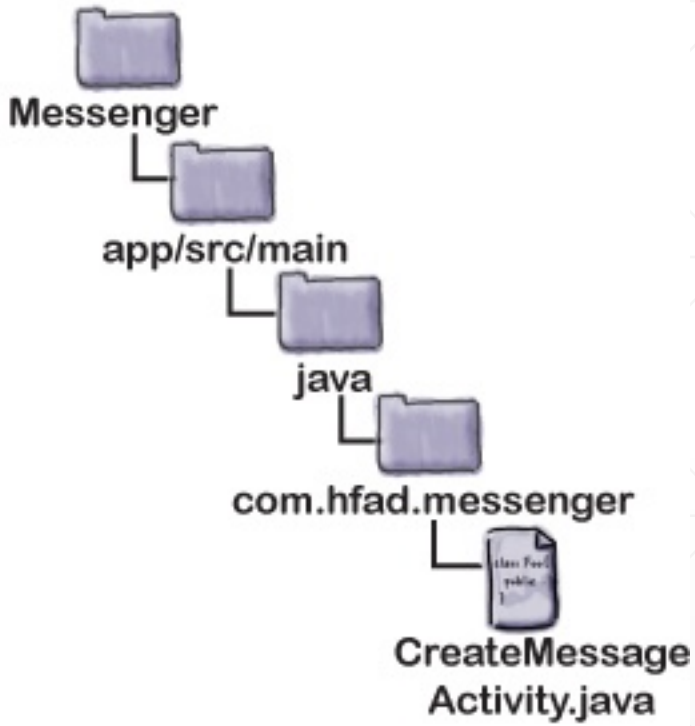
Add a new String called send. We gave ours a value of Send Message so that the text "Send Message" appears on the button.

```
package com.hfad.messenger;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;

public class CreateMessageActivity extends Activity {

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

    //Call onSendMessage() when the button is clicked
    public void onSendMessage(View view) {
    }
}
```

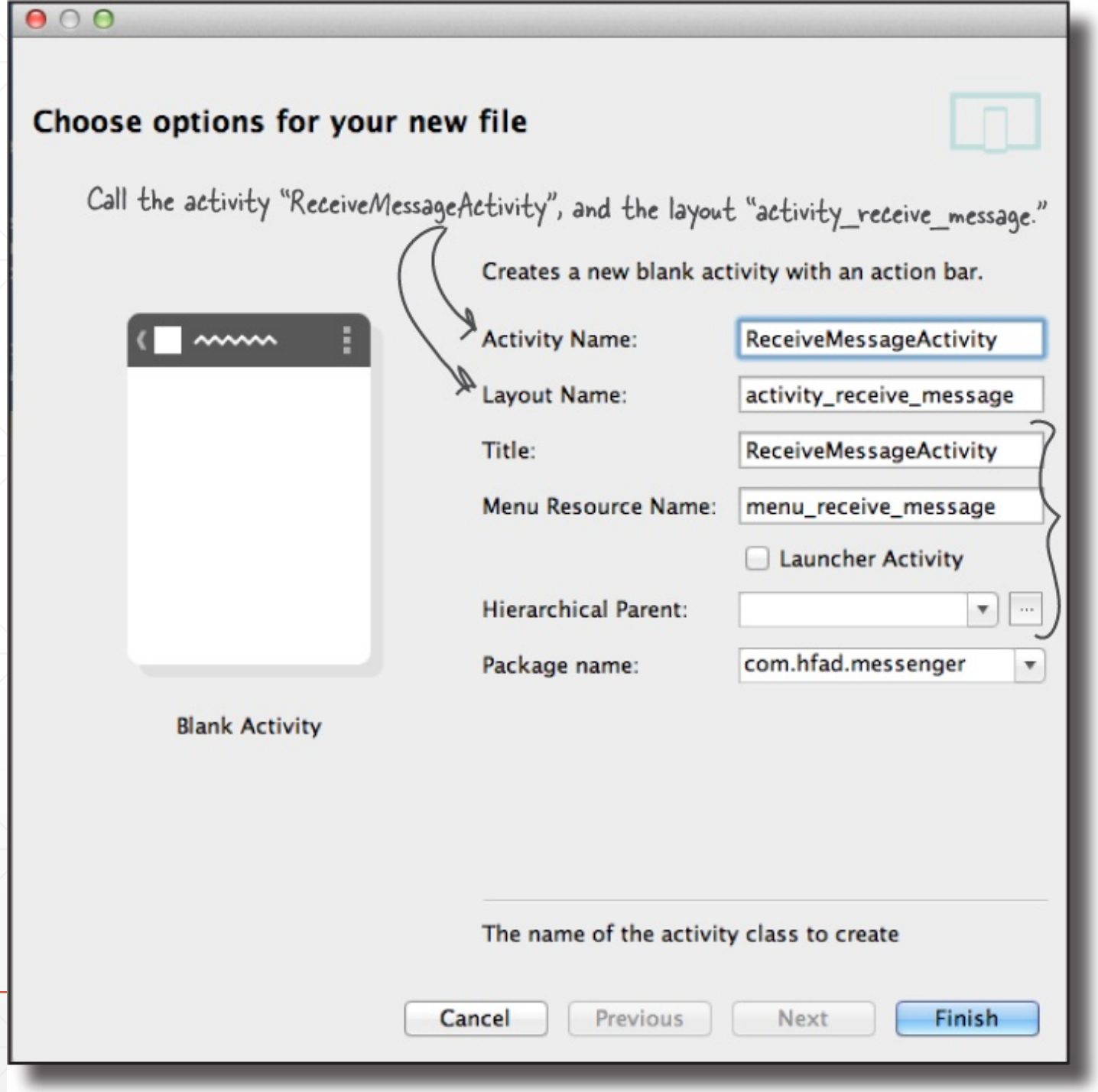
We're replacing the code that Android Studio created for us, as most of the code it creates isn't required.

The onCreate() method gets called when the activity is created.

This method will get called when the button's clicked. We'll complete the method body as we work our way through the rest of the chapter.



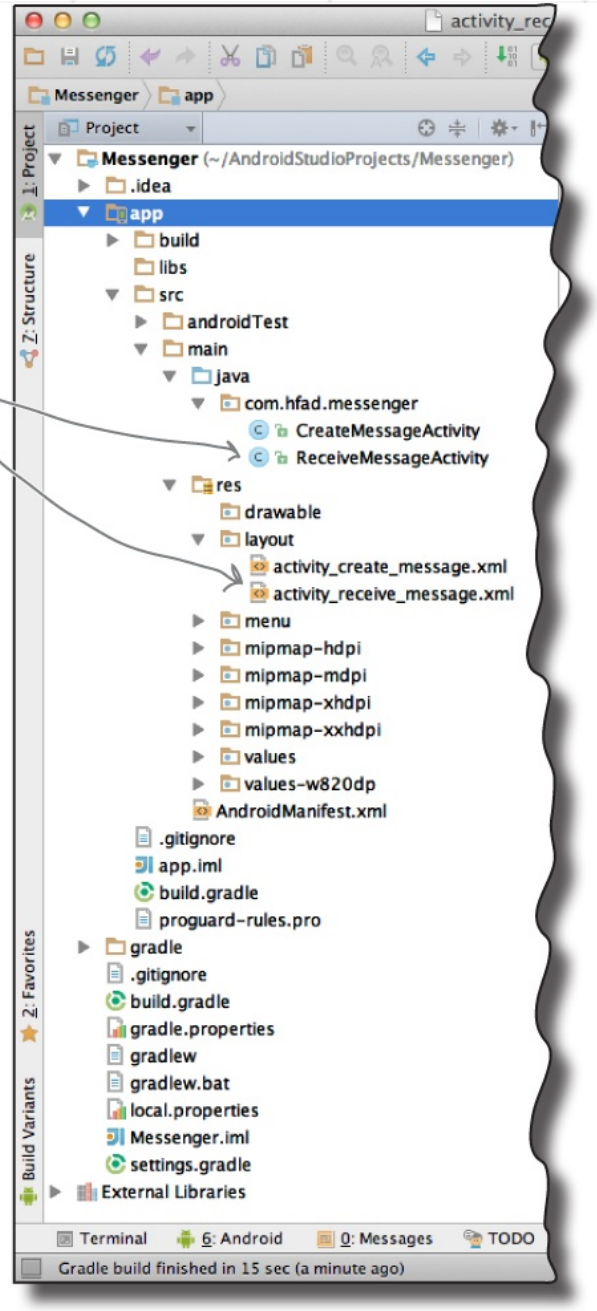
Create 2nd Activity



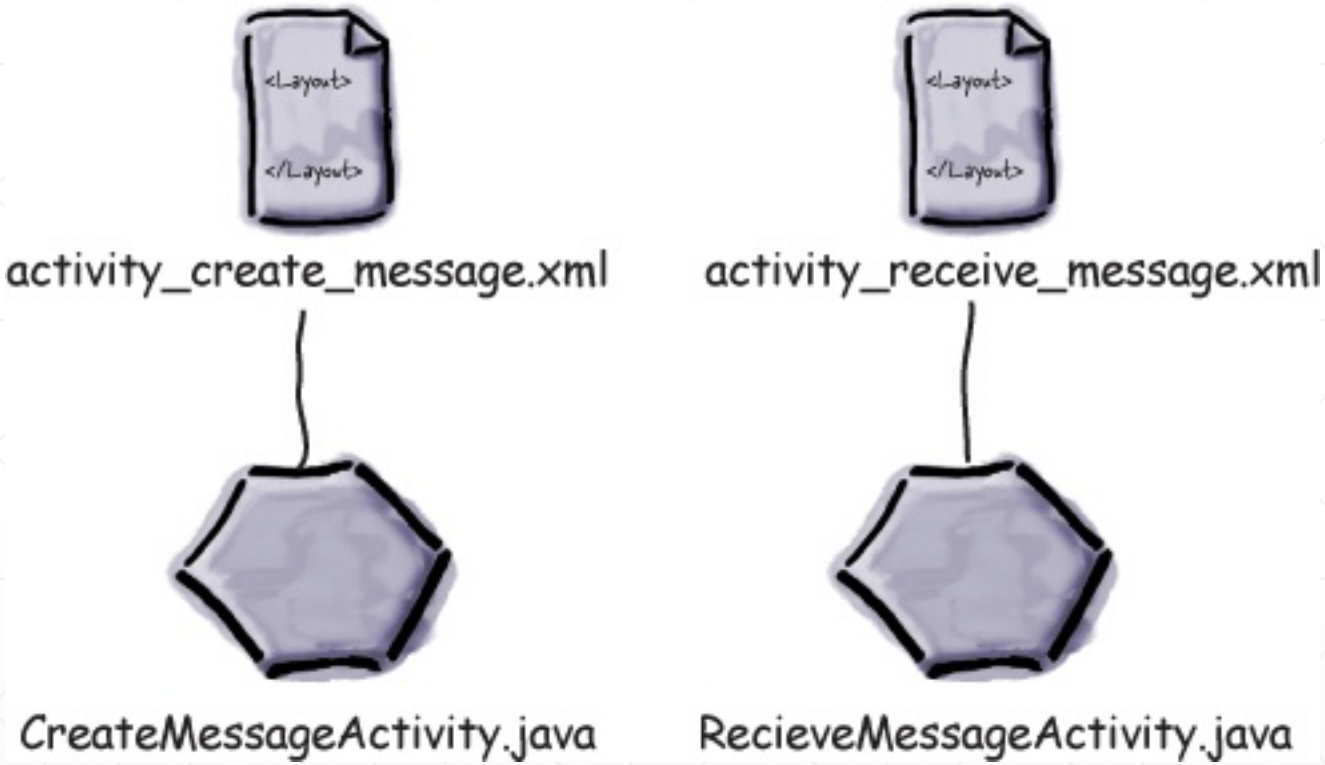
Accept the rest of the defaults, as all we're interested in is creating a new activity and layout. We'll replace most of the code Android Studio gives us.

Create 2nd Activity

Here's the new activity and layout we just created.
There are now two activities and layouts in the app.



Create 2nd Activity



Create 2nd Activity

You can find AndroidManifest.xml in this folder.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.hfad.messenger" >

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >

        <activity
            android:name=".CreateMessageActivity"
            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=".ReceiveMessageActivity"
            android:label="@string/title_activity_receive_message" >
        </activity>

    </application>

</manifest>
```

This is the package name we specified.

Android Studio gave our app a default icon. We'll look at this later in the book.

The theme affects the appearance of the app. We'll look at this later.

This is the first activity, Create Message Activity.

This bit specifies that it's the main activity of the app.

This says the activity can be used to launch the app.

This is the second activity, Receive Message Activity.

Android Studio added these lines for us when we added the second activity.

Create 2nd Activity

```
<application
    ...
    ...>
    <activity
        android:name="activity_class_name"
        android:label="@string/activity_label"
        ...
        ...>
    ...
    </activity>
    ...
</application>
```

Each activity needs to be declared inside the <application> element.

This line is mandatory.

This line is optional, but Android Studio completes it for us.

The activity may have other properties too.

Create 2nd Activity

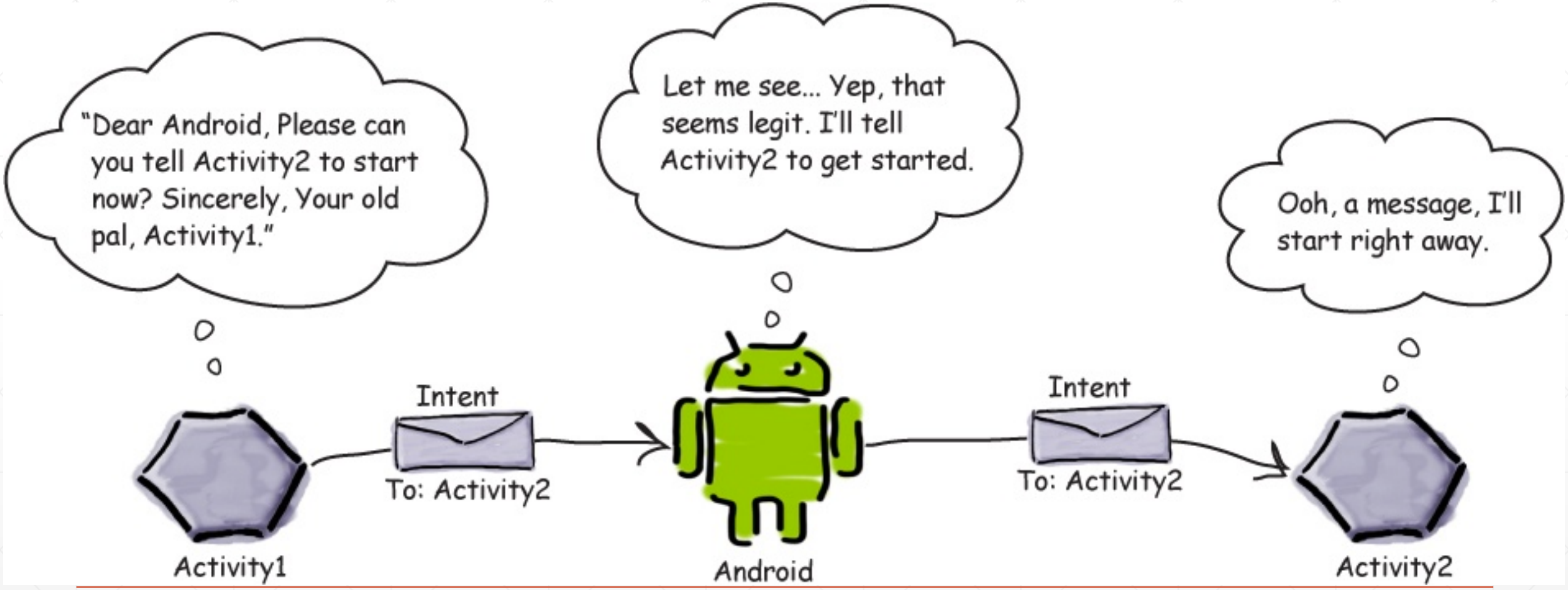
```
Intent intent = new Intent(this, Target.class);
```

Create 2nd Activity

```
startActivity(intent);
```

startActivity() starts the activity specified in the intent..

Create 2nd Activity

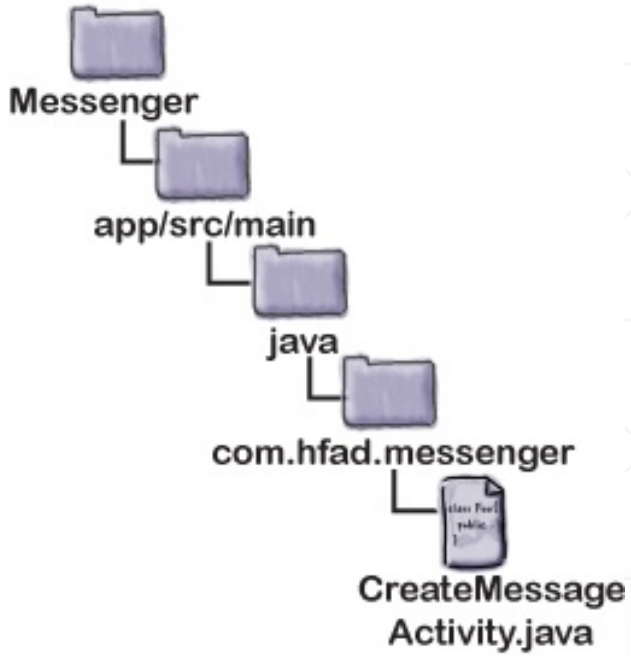


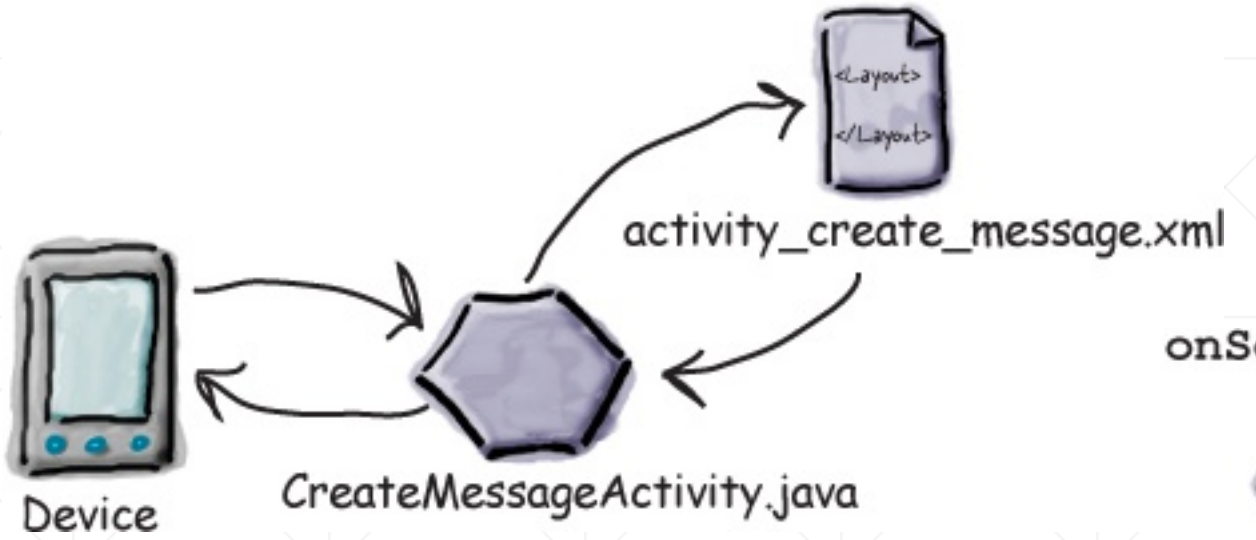
```
package com.hfad.messenger;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;

public class CreateMessageActivity extends Activity {

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

    //Call onSendMessage() when the button is clicked
    public void onSendMessage(View view) {
        Intent intent = new Intent(this, ReceiveMessageActivity.class);
        startActivity(intent);
    }

}
```

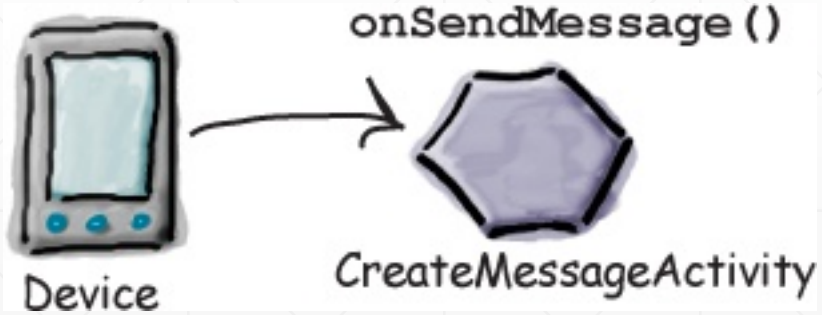
We need to import the Intent class android.content.Intent as we're using it in onSendMessage().

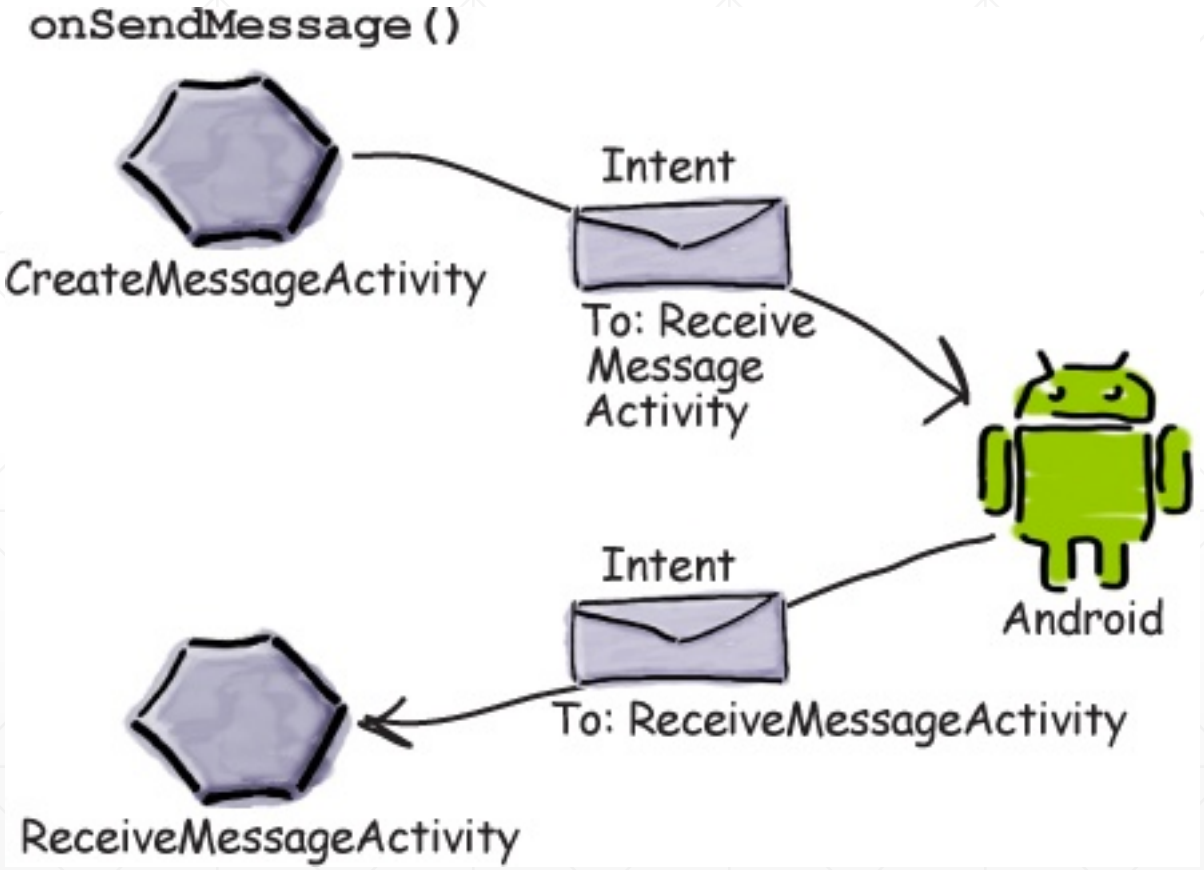
We've not changed this method.

Start activity ReceiveMessageActivity.

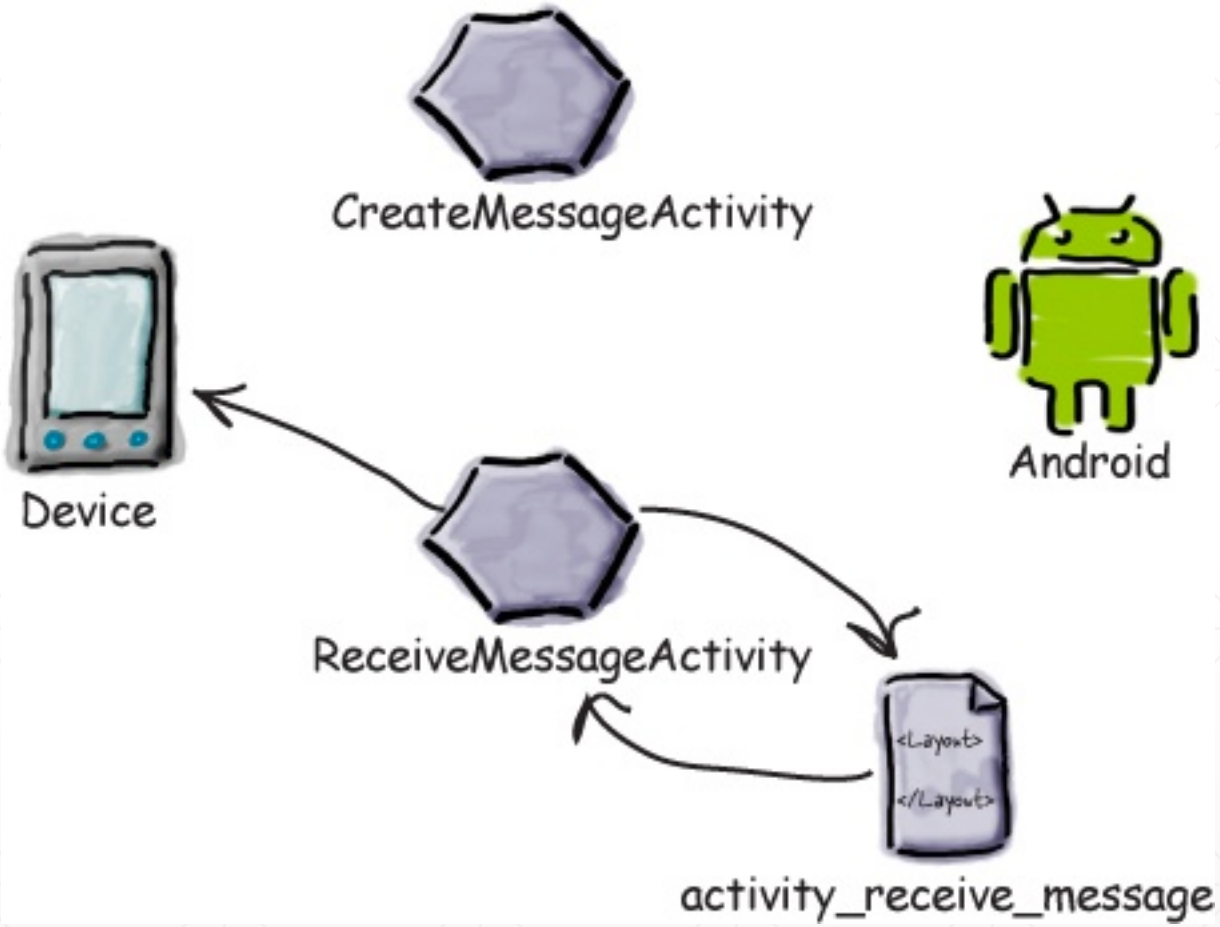



<Layout>
</Layout>
activity_create_message.xml
Device
CreateMessageActivity.java


onSendMessage()
Device
CreateMessageActivity

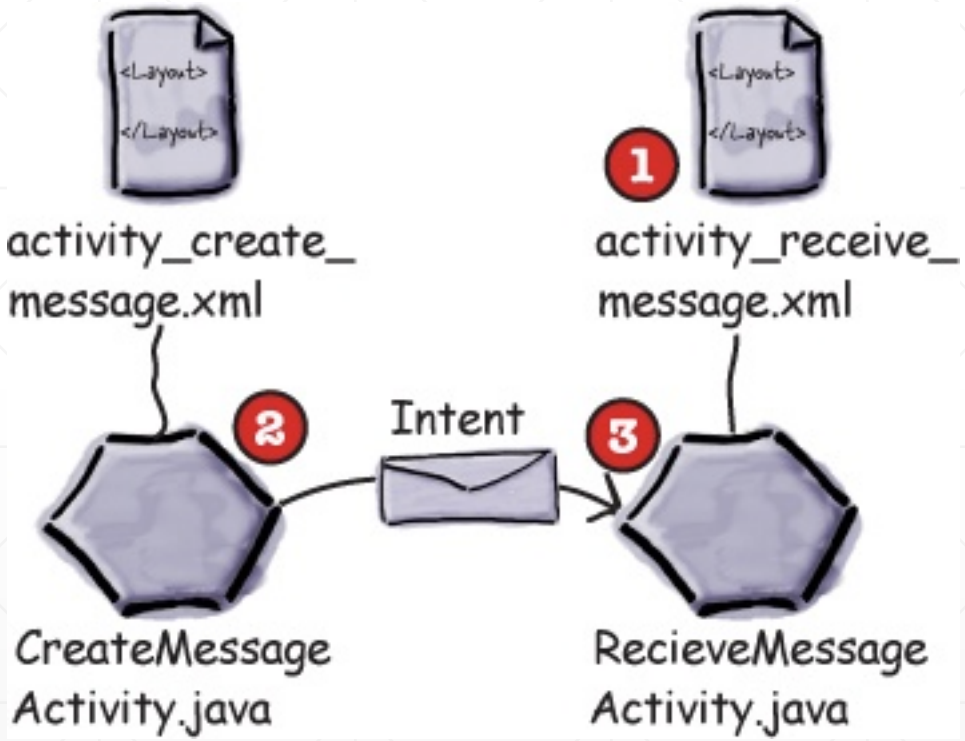

onSendMessage()
CreateMessageActivity
Intent
To: Receive
Message
Activity
Android
Intent
To: ReceiveMessageActivity
ReceiveMessageActivity

Call 2nd Activity



Call 2nd Activity

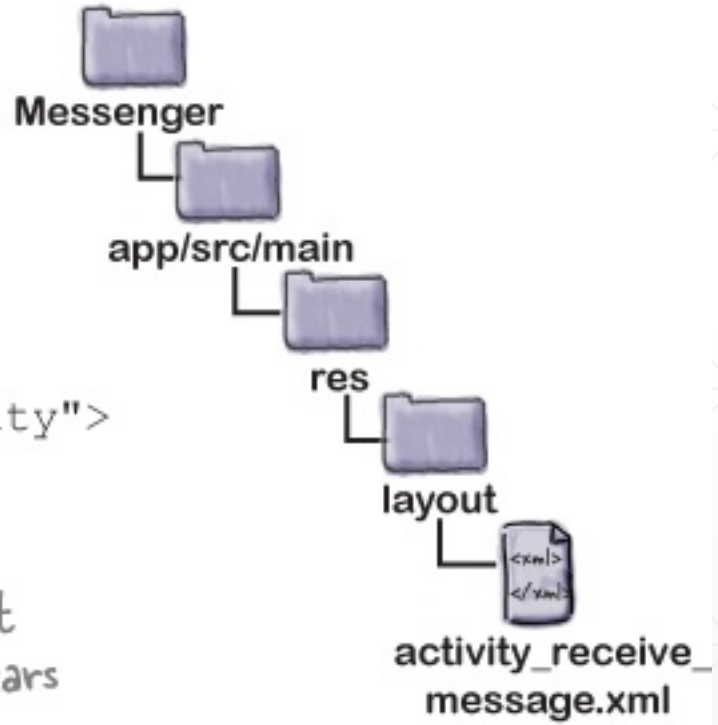
Pass Data


<Layout>
</Layout>
activity_create_
message.xml
<Layout>
</Layout>
1
activity_receive_
message.xml
2
Intent
3
CreateMessage
Activity.java
RecieveMessage
Activity.java

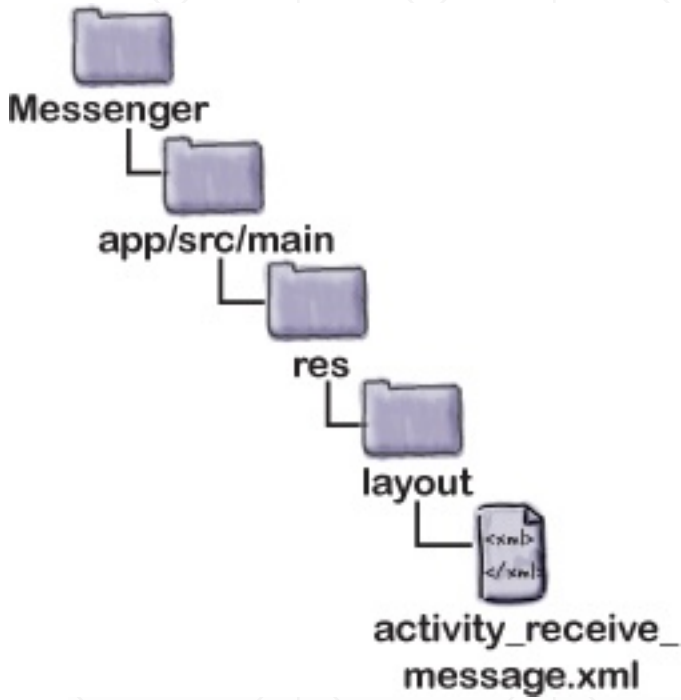
Pass Data

```
<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="com.hfad.messenger.ReceiveMessageActivity">

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

Here's the text view that currently appears in the layout.



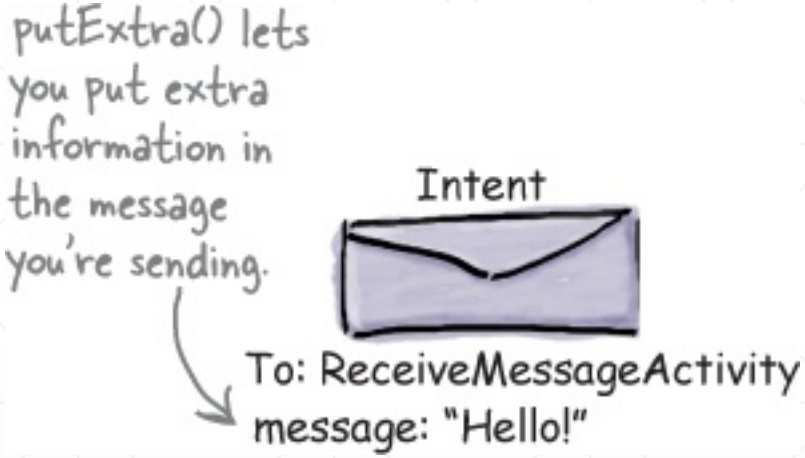
```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"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

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



This line gives the <TextView> an ID of message.

Remove the line that sets the text to @string/hello_world.

Pass Data – Send Data



intent.putExtra("message", value);

Pass Data – Receive Data

To: ReceiveMessageActivity
message: "Hello!"

```
Intent intent = getIntent();
String string = intent.getStringExtra("message");
```

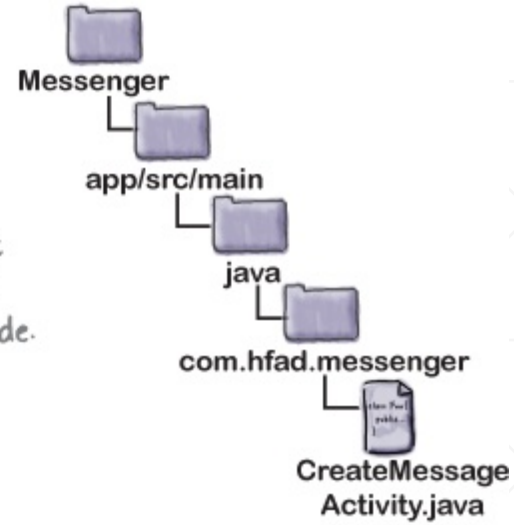
Get the intent.

Get the string passed along with the intent that has a name of "message".

Pass Data – Send Data

```
package com.hfad.messenger;

import android.os.Bundle;
import android.app.Activity;
import android.content.Intent;
import android.view.View;
import android.widget.EditText;

public class CreateMessageActivity extends Activity {

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

    //Call onSendMessage() when the button is clicked
    public void onSendMessage(View view) {
        EditText messageView = (EditText)findViewById(R.id.message);
        String messageText = messageView.getText().toString();
        Intent intent = new Intent(this, ReceiveMessageActivity.class);
        intent.putExtra(ReceiveMessageActivity.EXTRA_MESSAGE, messageText);
        startActivity(intent);
    }
}
```

You need to import the EditText class android.widget.EditText as you're using it in your activity code.



Get the text that's in the EditText.

Start ReceiveMessageActivity with the intent.

Create an intent, then add the text to the intent. We're using a constant for the name of the extra information so that we know CreateMessageActivity and ReceiveMessageActivity are using the same String. We'll add this to ReceiveMessageActivity on the next page.

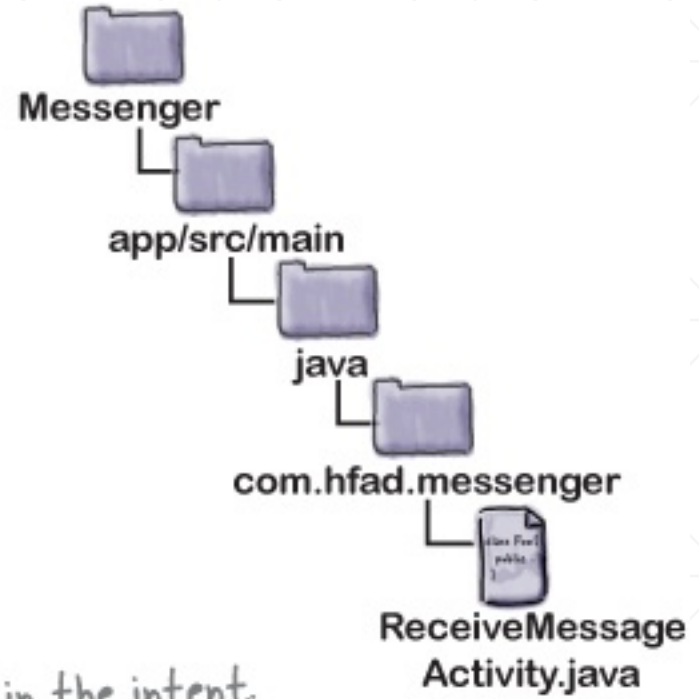
```
package com.hfad.messenger;

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

public class ReceiveMessageActivity extends Activity {

    public static final String EXTRA_MESSAGE = "message";

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

We need to import the Intent and TextView classes.

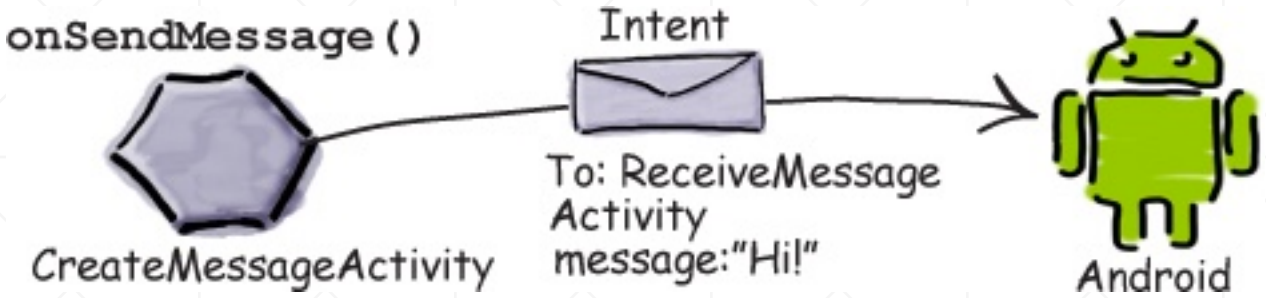
This is the name of the extra value we're passing in the intent.

Get the intent, and get the message from it using getStringExtra().

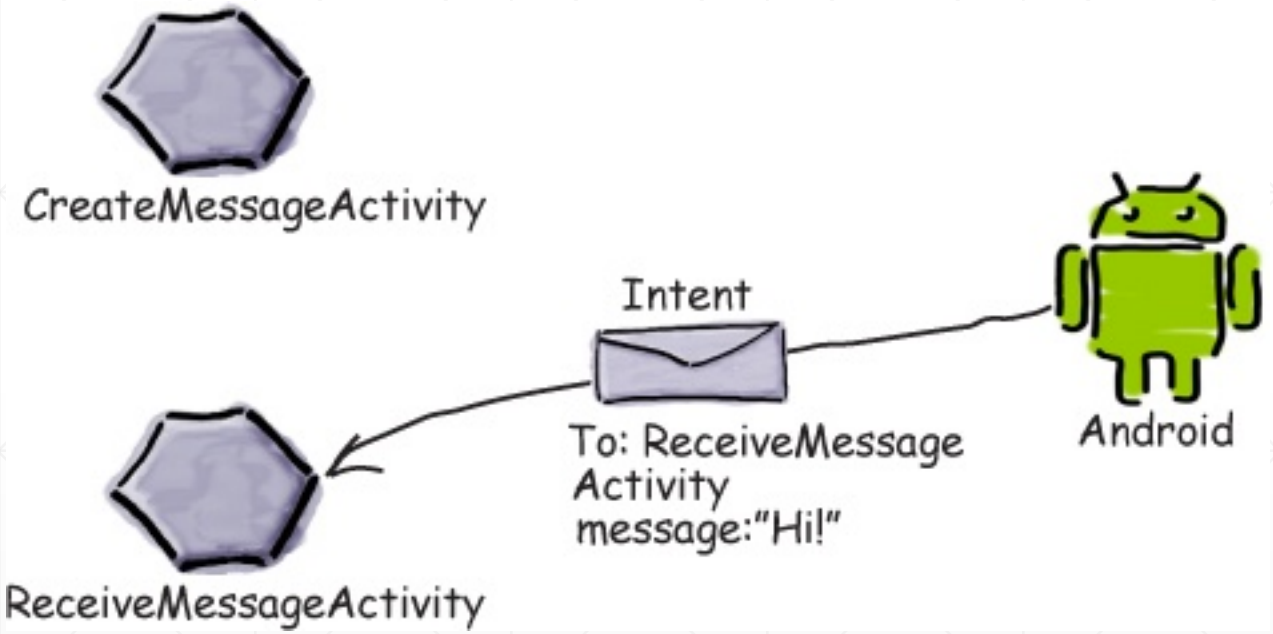
Add the text to the message text view.



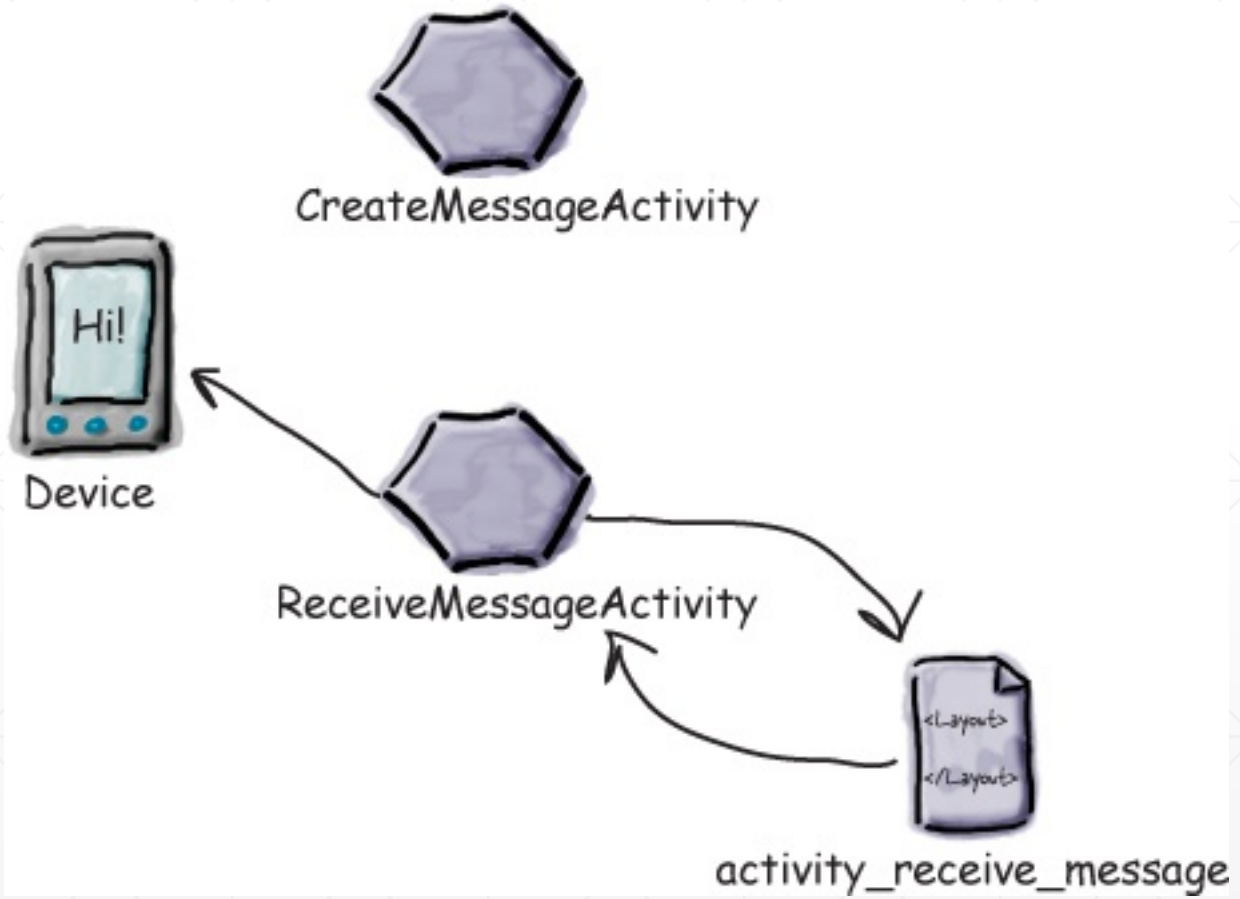
Multiple Activities - Recap



Multiple Activities - Recap



Multiple Activities - Recap



Multiple Activities

- A task is two or more activities chained together.
- The <EditText> element defines an editable text field for entering text. It inherits from the Android View class.
- You can add a new activity in Android Studio by choosing File → New... → Activity.
- Each activity you create must have an entry in AndroidManifest.xml.
- An intent is a type of message that Android components use to communicate with one another.
- An explicit intent explicitly specifies the component the intent is targeted at.
  You create an explicit intent using Intent intent = new Intent(this, Target.class);
- To start an activity, call startActivity(intent). If no activities are found, it throws an ActivityNotFoundException.
- Use the putExtra() method to add extra information to an intent.
- Use the getIntent() method to retrieve the intent that started the activity