Android Developer Fundamentals

Phone and SMS

Lesson 2

2 SMS Messages

Contents

- Implicit intent vs. SmsManager
- Formatting the phone number
- ACTION_SENDTO to use an installed messaging app
- Using SmsManager to send from within your app
- Receiving SMS messages

Implicit intent vs. SmsManager

- ACTION_SENDTO with implicit intent: Use installed messaging app
  - Pro: Simplest action without need for requesting user permission
  - Con: User navigates back to your app from the messaging app
- SmsManager: Send message from within your app
  - Pro: Keeps user within your app
  - Con: Need to request user permission

Phone number URI for sending SMS

Prepare a Uniform Resource Identifier (URI) as a string

- editText: EditText for entering a phone number (`14155551212`)
- smsNumber: String prefixed by "smsto:", e.g. (`smsto:14155551212`)

```
String smsNumber =
    String.format("smsto: %s", editText.getText().toString());
```

PhoneNumberUtils class

PhoneNumberUtils class provides utility methods for phone number strings

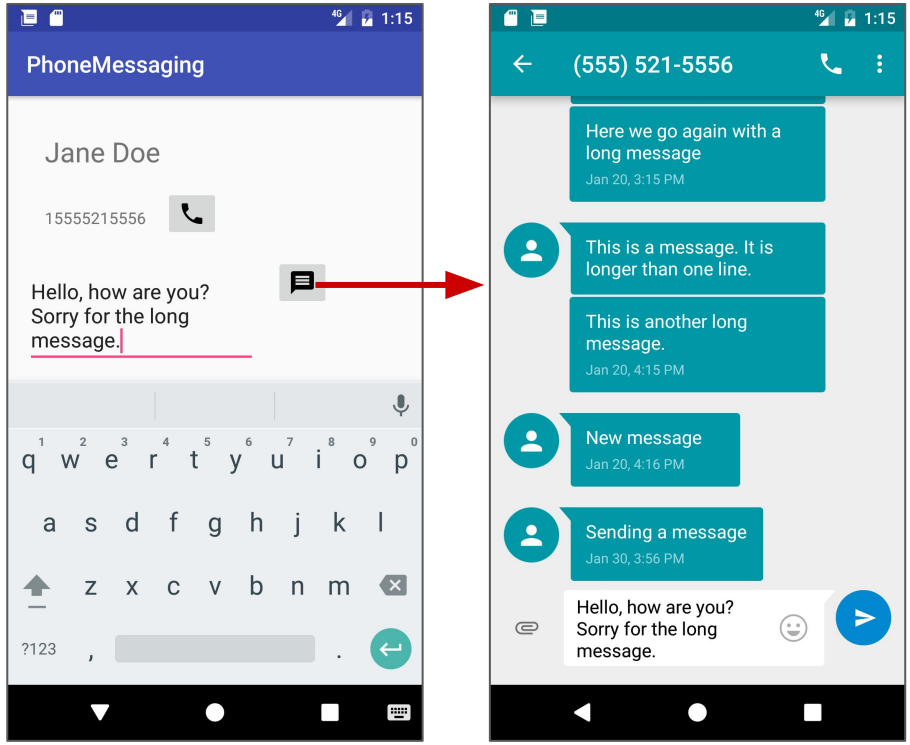
- normalizeNumber() removes characters other than digits
- formatNumber() formats a phone number for a specific country if the number doesn't already include a country code

Intent with ACTION_SENDTO

Use a messaging app for sending

ACTION_SENDTO: Use an installed messaging app to send the message

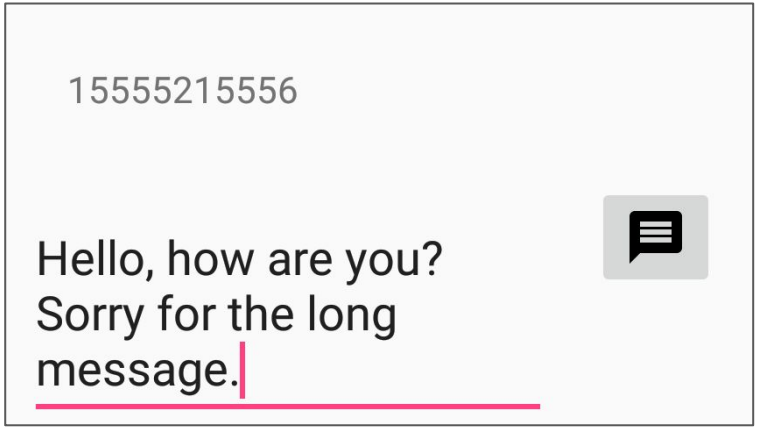
- User can change phone number and message before sending
- User navigates back to your app by tapping **Back**



Add onClick handler for send button

1. Add a send message button, phone number, and message
2. Add `android:onClick` attribute to button
3. Create handler for onClick

```
public void smsSendMessage(View view) {
    ...
}
```



Create intent with phone and message

1. Prepare a URI

```
String smsNumber =
    String.format("smsto: %s", editText.getText().toString());
```

2. Create implicit Intent

```
Intent smsIntent = new Intent(Intent.ACTION_SENDTO);
```

3. setData() with phone number

```
smsIntent.setData(Uri.parse(smsNumber));
```

4. putExtra() with message string sms and key "sms_body"

```
smsIntent.putExtra("sms_body", sms);
```

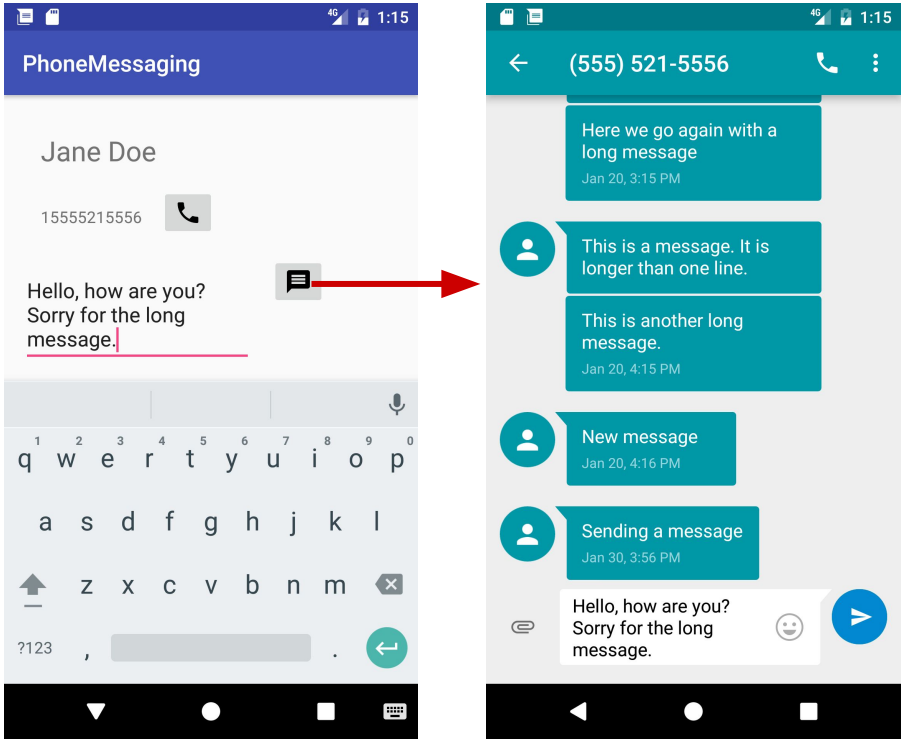
Send the intent to a messaging app

- Use resolveActivity() with getPackageManager() to resolve the implicit intent to an installed app
- If result non-null, call `startActivity()`

```
if (smsIntent.resolveActivity(getPackageManager()) != null) {
    startActivity(smsIntent);
} else {
    ... // Log and show explanation
```

Messaging app sends the message

- Enter a message
- Tap button to send message
- Phone number and message appears in messaging app



SmsManager

Using SmsManager

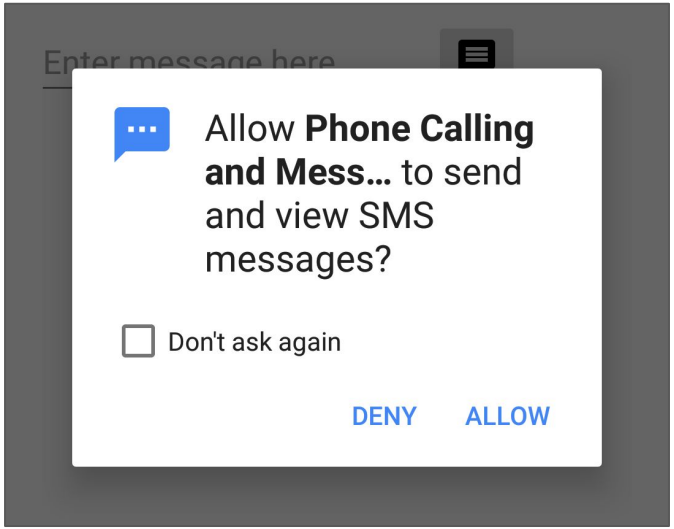
1. Add permissions to AndroidManifest.xml

```
<uses-permission android:name="android.permission.SEND_SMS" />
```

2. Check if the user still grants SMS permission, or request it if needed
3. Use the sendTextMessage() method

Do you have SMS permission?

- If the user turned off SMS permission for the app, your app can request permission



Check permission steps

- Define integer to use for the `requestCode` parameter

```
private static final int MY_PERMISSIONS_REQUEST_SEND_SMS = 1;
```

- Use checkSelfPermission()
- Use requestPermissions()
  - Context (`this`), and `SEND_SMS` in the string array of permissions
  - The request integer constant `MY_PERMISSIONS_REQUEST_SEND_SMS`

Check permission code

```
if (ActivityCompat.checkSelfPermission(this,
                    Manifest.permission.SEND_SMS) !=
                    PackageManager.PERMISSION_GRANTED) {
    // Permission not yet granted. Use requestPermissions()
    ActivityCompat.requestPermissions(this,
                    new String[]{Manifest.permission.SEND_SMS},
                    MY_PERMISSIONS_REQUEST_SEND_SMS);
```

Get the user's response

Override onRequestPermissionsResult() to check if returned `requestCode` is `MY_PERMISSIONS_REQUEST_SEND_SMS`

```
@Override
public void onRequestPermissionsResult(int requestCode,
                       String permissions[], int[] grantResults) {
    switch (requestCode) {
        case MY_PERMISSIONS_REQUEST_SEND_SMS: {
            ...
```

Check if request was granted

- Response returned in `permissions` array (index 0 if only a single request was made)
- Compare to grant result: `PERMISSION_GRANTED` or `PERMISSION_DENIED`.

```
if (permissions[0].equalsIgnoreCase(Manifest.permission.SEND_SMS)
        && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
    // Permission was granted.
} else {
    ...
```

Using SmsManager to send

Use sendTextMessage() with parameters

- `destinationAddress`: String for phone number
- `scAddress`: String for the SMSC (service center address), or `null` for current default; usually pre-set in the SIM card
- `smsMessage`: String for message body
- `sentIntent`: PendingIntent broadcast when the message is successfully sent or if the message failed, or use `null`
- `deliveryIntent`: PendingIntent broadcast when the message is delivered to the recipient, or use `null`

Steps to send an SMS message (1)

1. Declare string and PendingIntent parameters

```
 ...
// Set the service center address if needed, otherwise null.
String scAddress = null;
// Set pending intents to broadcast
// when message sent and when delivered, or set to null.
PendingIntent sentIntent = null, deliveryIntent = null;
...
```

Steps to send an SMS message (2)

2. Create an smsManager
3. Use sendTextMessage()

```
SmsManager smsManager = SmsManager.getDefault();
smsManager.sendTextMessage
           (destinationAddress, scAddress, smsMessage,
                               sentIntent, deliveryIntent);
```

Receiving SMS Messages

Using a broadcast receiver

1. Add permissions to AndroidManifest.xml

```
<uses-permission android:name="android.permission.RECEIVE_SMS" />
```

2. Add BroadcastReceiver
3. Register broadcast receiver for the `SMS_RECEIVED` intent
4. Override onReceive() method of broadcast receiver

Add a broadcast receiver

1. Select the package name
2. Choose **File > New > Other > Broadcast Receiver**
    - Make sure "Exported" and "Enabled" are checked
    - Android Studio generates tags in AndroidManifest.xml

```
<receiver
android:name="com.example.android.smsmessaging.MySmsReceiver"
    android:enabled="true"
    android:exported="true">
</receiver>
```

Register the broadcast receiver

Register receiver for intent android.provider.Telephony.SMS_RECEIVED

```
<receiver
  android:name="com.example.android.smsmessaging.MySmsReceiver"
    android:enabled="true"
    android:exported="true">
    <intent-filter>
        <action
           android:name="android.provider.Telephony.SMS_RECEIVED"/>
    </intent-filter>
</receiver>
```

Override receivers onReceive()

1. Get SMS message: retrieve intent extras, store in `bundle`, and define `msgs` array
2. Get the `format` to use with createFromPdu() to create SmsMessage
3. Use createFromPdu() to fill `msgs` array
4. Get originating address using `getOriginatingAddress()`
5. Get message body using `getMessageBody()`

1. Get SMS message

Retrieve intent extras, store in a `bundle`, and define `msgs` array

```
@Override
public void onReceive(Context context, Intent intent) {
    // Get the SMS message.
    Bundle bundle = intent.getExtras();
    SmsMessage[] msgs;
    String strMessage = "";
    String format = bundle.getString("format");
```

2. Get the `format`

- Use createFromPdu() to create the SmsMessage, using the `format` provided in the bundle
- The `format` is from an `SMS_RECEIVED_ACTION` broadcast
  - "3gpp" for GSM/UMTS/LTE messages in the 3GPP format, or
  - "3gpp2" for CDMA/LTE messages in 3GPP2 format

```
String format = bundle.getString("format");
```

3. Use createFromPdu()

Use createFromPdu() to fill the `msgs` array

- Android version 6.0 (Marshmallow) and newer:
  createFromPdu(byte[] pdu, String format)

```
msgs[i] = SmsMessage.createFromPdu((byte[]) pdus[i], format);
```

- Earlier versions of Android:
  createFromPdu(byte[] pdu)

```
msgs[i] = SmsMessage.createFromPdu((byte[]) pdus[i]);
```

4. Get originating address

Get originating address using `getOriginatingAddress()`

```
strMessage += "SMS from " + msgs[i].getOriginatingAddress();
```

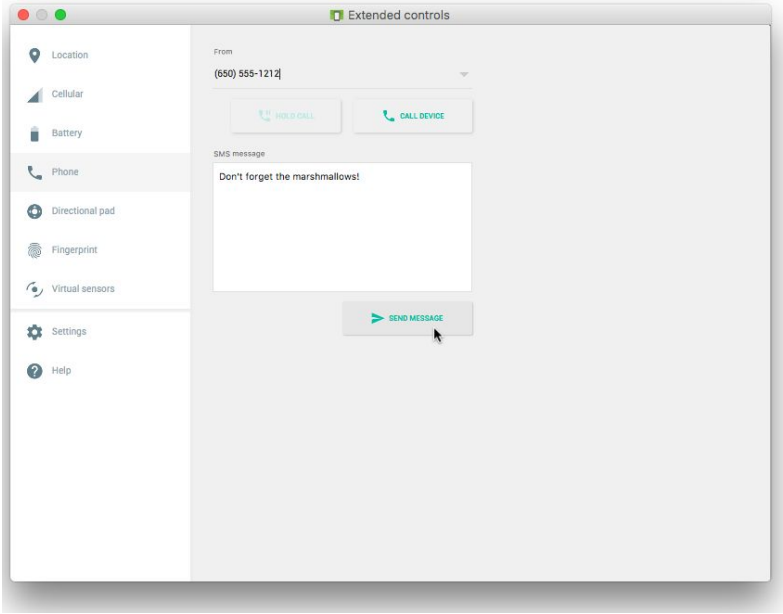
5. Get message body

Get message body using `getMessageBody()`

```
strMessage += " :" + msgs[i].getMessageBody() + "\n";
```

Receiving messages in the emulator

1. Run app on emulator
2. Click the … (More) icon
3. Click **Phone** in the left column
4. Enter a message
5. Click **Send Message**



Learn more

- Common Intents and Intents and Intent Filters
- SmsManager
- SmsMessage
- BroadcastReceiver
- createFromPdu()
- Requesting Permissions at Run Time
- checkSelfPermission
- Simulating incoming call or sms in Android Studio
- Getting Your SMS Apps Ready for KitKat

What's Next?

- Concept Chapter: SMS Messages
- Practical: Sending and Receiving SMS Messages

END