tuts+   Subscribe   Sign In   ☰

CODE > ANDROID SDK

</> Android Essentials: Creating Simple User Forms

by Shane Conder & Lauren Darcey   22   Difficulty: Beginner   Length: Medium   Languages: English ▼
Jul 2010

Android SDK   Mobile Development   💬 🔗

This post is part of a series called Android Essentials.
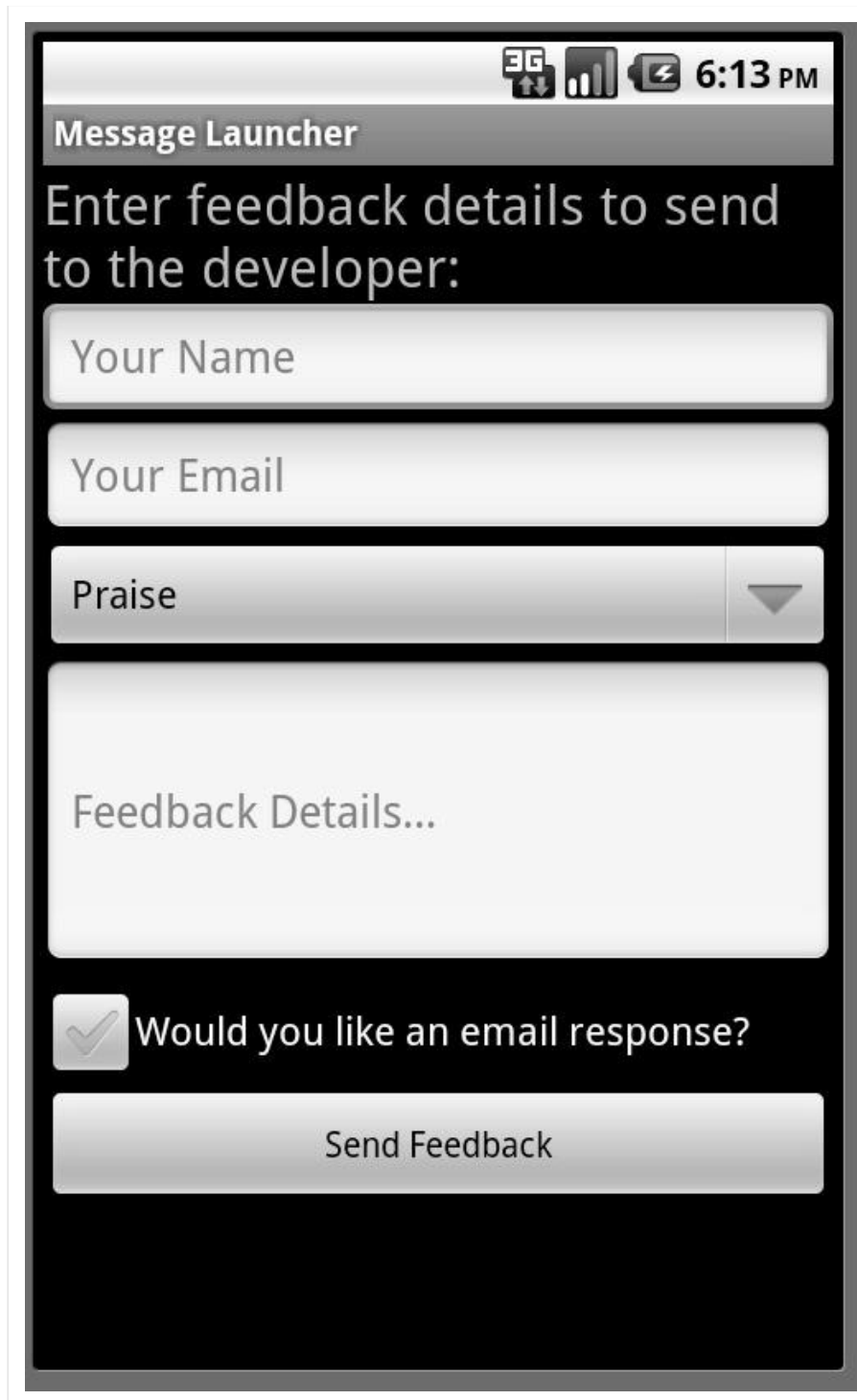
⏩   Android Essentials: Using the Contact Picker

Android applications often rely upon data supplied by users. This tutorial walks you through the creation and use of a number of the most common controls used to collect data from the user, including:

- The EditText control
- The Spinner control
- The Checkbox control
- The Button control

By the way, if you're looking for a quick solution, check out Android Dynamic Form, a dynamic form generator plugin available on Envato Market.

Android Dynamic Form

For this tutorial, you design and implement a form within your Android application which allows the user to supply important feedback to the developer. The user is given a number of options for submitting different types of feedback. This feedback can then be sent to the developer as an email. The form you create will ultimately look like this:

Assumptions

The authors are assuming the reader has some basic knowledge of Android and have all of the tools installed and working—specifically Eclipse, the Android SDK and the Android ADT plug-in for Eclipse.

Readers may also benefit from reading our Quick Tip: Enabling Users to Send Email From Your Android Applications – The Easy Way.

Step 0: Creating a Simple Android Project

Begin by creating a new Android project. You can also follow along using the source code provided as a supplement to this tutorial.

Advertisement

Step 1: Designing the Form

First, you need to give some thought to want kind of data you want to collect from the user. The form may have any number of fields. Consider the types of data you want to collect and choose the appropriate type of control. For example:

- To collect text input, use EditText controls
- To limit the user to a fixed set of responses, use Spinner controls, similar to a drop-down menu
- To collect boolean (yes/no) input, use CheckBox controls
- To allow the user to trigger events, use Button controls

For this tutorial, you will be designing a feedback form. This form collects five pieces of data from the user:

- The user's name (a string)
- The user's email (a string)
- The type of feedback (options: Praise, Gripe, Suggestion or Bug)
- The feedback message (a string)
- Whether or not the user wants an email response (a boolean)

Step 2: Creating the Layout Resource

Begin by creating a layout resource for the form screen. The form will have a bunch of fields, which could span more than a single screen (depending on the device screen size), so you should consider wrapping the entire form within a ScrollView control to enable scrollbars.

The ScrollView control must have exactly one child view, so consider which layout control is most appropriate for the form you want to create. Forms are often contained within a vertically oriented LinearLayout control, so that the

form fields cascade down the page vertically, one after another. This also helps the user's focus move from field to field naturally.

A simple form layout resource might look like this:

```
01   <?xml version="1.0" encoding="utf-8"?>
02   <ScrollView
03       xmlns:android="http://schemas.android.com/apk/res/android"
04       android:id="@+id/ScrollView01"
05       android:layout_width="wrap_content"
06       android:layout_height="wrap_content"
07       android:scrollbars="vertical">
08       <LinearLayout
09           android:layout_width="fill_parent"
10           android:orientation="vertical"
11           android:layout_height="fill_parent">
12
13   <!--Put form controls here-->
14
15       </LinearLayout>
16   </ScrollView>
```

Step 3: Add a TextView Control (Form Description)

Next, you need to add a TextView control within the LinearLayout control. The TextView control called TextViewTitle displays the form description and purpose to the user. This control displays a string resource called @string/feedbacktitle, which must be defined within the /res/values/strings.xml string resource file.

Here is the XML to add to your form layout resource file:

```
1   <TextView
2       android:id="@+id/TextViewTitle"
3
```

```
4         android:layout_width="wrap_content"
5         android:layout_height="wrap_content"
6         android:text="@string/feedbacktitle"
7         android:textSize="10pt">
</TextView>
```

Step 4: Add an EditText Control (Name)

Now you need to add your first EditText control just below the TextView control you just created. This EditText control called EditTextName acts as a form field for the user's name. You can use the hint attribute to supply a string to display in the EditText control when it's empty (e.g. "Type your name here…"). You can also set the inputType attribute of the EditText control to apply name entering logic.

Here is the XML to add to your form layout resource file:

```
1   <EditText
2       android:id="@+id/EditTextName"
3       android:layout_height="wrap_content"
4       android:hint="@string/feedbackname"
5       android:inputType="textPersonName"
6       android:layout_width="fill_parent">
7   </EditText>
```

Step 5: Add another EditText Control (Email)

Next, you need to add your second EditText control just below the EditText control called EditTextName. This EditText control called EditTextEmail acts as a form field for the user's email address. Again, set the hint attribute to supply a string to display in the EditText control when it's empty. This time, set the

inputType attribute of the EditText control to textEmailAddress, which will make entering emails easier on the user.

Here is the XML to add to your form layout resource file:

```
1   <EditText
2       android:id="@+id/EditTextEmail"
3       android:layout_height="wrap_content"
4       android:hint="@string/feedbackemail"
5       android:inputType="textEmailAddress"
6       android:layout_width="fill_parent">
7   </EditText>
```

Step 6: Add a Spinner Control (Feedback Type)

Next, you need to add a Spinner control just below the EditText control you just created. This Spinner control called SpinnerFeedbackType allows the user to select the type of feedback from a fixed list of options (Praise, Gripe, Suggestion, or Bug).

First, you need to define these choices as individual string resources in the strings.xml resource file.

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <resources>
3       <!--Other string resources also defined in this file… -->
4       <string name="feedbacktype1">Praise</string>
5       <string name="feedbacktype2">Gripe</string>
6       <string name="feedbacktype3">Suggestion</string>
7       <string name="feedbacktype4">Bug</string>
8   </resources>
```

Next, create a string array resource using the individual string resources as follows in /res/values/arrays.xml:

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <resources>
3       <string-array name="feedbacktypelist">
4           <item>@string/feedbacktype1</item>
5           <item>@string/feedbacktype2</item>
6           <item>@string/feedbacktype3</item>
7           <item>@string/feedbacktype4</item>
8       </string-array>
9   </resources>
```

Now you are ready to configure the Spinner control in your form layout. Begin by supplying the prompt attribute, which will provide a helpful string at the top of the Spinner control. Next, specify the list of string choices using the entries attribute—specifically, set the entries attribute to the string array you just defined: @array/feedbacktypelist.

Here is the XML to add to your form layout resource file:

```
1   <Spinner
2       android:id="@+id/SpinnerFeedbackType"
3       android:layout_height="wrap_content"
4       android:prompt="@string/feedbacktype"
5       android:layout_width="fill_parent"
6       android:entries="@array/feedbacktypelist">
7   </Spinner>
```

Step 7: Add a Multi-Line EditText Control (Feedback)

Next, you need to add one more EditText control just below the Spinner control. This EditText control called EditTextFeedbackBody acts as a form field for the

feedback text. Again, set the hint attribute to supply a string to display in the EditText control when it's empty. This time you want to give the user ample space to write praise, gripes, suggestions, or describe bugs in the application. Therefore, you may want to set the inputType attribute of the EditText control to textMultiLine and specify the number of lines to draw using the lines attribute.

Here is the XML to add to your form layout resource file:

```
1   <EditText
2       android:id="@+id/EditTextFeedbackBody"
3       android:layout_height="wrap_content"
4       android:hint="@string/feedbackbody"
5       android:inputType="textMultiLine"
6       android:lines="5"
7       android:layout_width="fill_parent">
8   </EditText>
```

Step 8: Add a CheckBox Control

Next, you need to add a CheckBox control just below the EditText control you just created. This CheckBox control called CheckBoxResponse allows the user to choose whether or not they want to request an email response from the app developer. You can use the text attribute to supply a string to display next to the CheckBox control.

Here is the XML to add to your form layout resource file:

```
1   <CheckBox
2       android:id="@+id/CheckBoxResponse"
3       android:layout_height="wrap_content"
4       android:text="@string/feedbackresponse"
5       android:layout_width="fill_parent">
6   </CheckBox>
```

Step 9: Add a Button Control

Finally, you are ready to finish off the form with a Button control. If you want to have a button with text on it, use the Button control; if you prefer a button with a picture on it, use an ImageButton control instead. We will use a Button control here. First, set the text on the Button control using the text attribute. Next, you can easily register a click handler (as opposed to registering it programmatically in your Activity) for your Button control using the onClick attribute.

Here is the XML to add to your form layout resource file:

```
1    <Button
2        android:id="@+id/ButtonSendFeedback"
3        android:layout_height="wrap_content"
4        android:text="@string/feedbackbutton"
5        android:onClick="sendFeedback"
6        android:layout_width="fill_parent">
7    </Button>
```

Excellent! You've finished designing your form. Now, all you need to do is implement the sendFeedback() method in your Activity.

Step 10: Implement a Button click handler

In the Button control, you specified the onClick attribute as sendFeedback. Now you will need to implement a method called sendFeedback() within your

Activity class. For example:

```
public void sendFeedback(View button) {
    // Do click handling here
}
```

Step 11: Reading Input from EditText Controls

Now that your form is designed and the controls have been implemented, you next need to collect the form data from the individual fields when the Button control is clicked.

For an EditText control, you use the getText() method.

```
final EditText nameField = (EditText) findViewById(R.id.EditText
String name = nameField.getText().toString();


final EditText emailField = (EditText) findViewById(R.id.EditTex
String email = emailField.getText().toString();


final EditText feedbackField = (EditText) findViewById(R.id.Edit
String feedback = feedbackField.getText().toString();
```

Step 12: Reading Input From Spinner Controls

Your form included a Spinner control. You use the getSelectedItem() method to read the data from this form control.

```
final Spinner feedbackSpinner = (Spinner) findViewById(R.id.Spin
String feedbackType = feedbackSpinner.getSelectedItem().toString
```

In this case, the selected item in the Spinner control is the String chosen by the user of the selected item.

Step 13: Reading Input from CheckBox Controls

Finally, your form included a CheckBox control. In this case, the result is just a flag to tell your application if the box was checked or not.

```
final CheckBox responseCheckbox = (CheckBox) findViewById(R.id.C
boolean bRequiresResponse = responseCheckbox.isChecked();
```

You can use this Boolean value however you want in your app.

Step 14: Generate the Appropriate Email Details

Now that you've got all your form data, you're ready to craft a message. Simply process all the data fields and build an appropriate feedback message. For example, you might use some fields in the message subject, and others in the message body. You can use format strings to help build the appropriate strings, the specifics of which will be discussed in an upcoming quick tip.

Conclusion

In this tutorial, you learned how to use various types of input controls to design a feedback form within an Android application. The EditText control is versatile and powerful, allowing for many different types of text and freeform input. The Spinner and Checkbox controls help limit the user's input to a specific set of responses. The Button control is a simple way to generate an event to process the form input.

There are many other controls worth exploring for use within forms. There is a lot more we could cover regarding good form design, how form controls fit into the Activity lifecycle, and how input methods and such factor into things, but for now, focus on gaining a good handle on the basics of form controls and how to use them.

We hope you enjoyed this tutorial and look forward to your feedback!

About the Authors

Mobile developers Lauren Darcey and Shane Conder have coauthored several books on Android development: an in-depth programming book entitled *Android Wireless Application Development* and *Sams TeachYourself Android Application Development in 24 Hours*. When not writing, they spend their time developing mobile software at their company and providing consulting services. They can be reached at via email to androidwirelessdev+mt@gmail.com, via their blog at androidbook.blogspot.com, and on Twitter @androidwireless.

Shane
Conder &
Lauren
Darcey

Shane Conder & Lauren Darcey

Mobile developers Lauren Darcey and Shane Conder have coauthored
numerous books on Android development. Our latest books include
Sams Teach Yourself Android Application Development in 24 Hours (3rd
Edition), Introduction to Android Application Development: Android
Essentials (4th Edition), and Advanced Android Application
Development (4th Edition). Lauren and Shane run a boutique consulting
firm specializing in the development of commercial-grade Android
applications for smartphones, tablets, wearables (i.e. Google Glass),
and more. They can be reached on Google+, their blog at
androidbook.blogspot.com and on Twitter @androidwireless.

Weekly email summary

Subscribe below and we'll send you a weekly email summary of all new Code tutorials. Never miss out on learning about the next big thing.

Email Address

Update me weekly

Translations

Envato Tuts+ tutorials are translated into other languages by our community members—you can be involved too!

Translate this post

Powered by

native

64 Comments   **Mobiletuts+**   **1** **Login**

♡ **Recommend** **3**   🐦 Tweet   f Share   Sort by Best

Join the discussion…

LOG IN WITH   **OR SIGN UP WITH DISQUS** (?)

                                          Name

Load more comments

✉ **Subscribe**   Ⓓ **Add Disqus to your siteAdd DisqusAdd**

🔒 **Disqus' Privacy PolicyPrivacy PolicyPrivacy**

LOOKING FOR SOMETHING TO HELP KICK START YOUR NEXT PROJECT?

Envato Market has a range of items for sale to help get you started.

WordPress Plugins

From $5

PHP Scripts

From $5

Unlimited Downloads
From $16.50/month

Get access to over one million creative
assets on Envato Elements.

Over 9 Million Digital Assets

Everything you need for your next
creative project.

QUICK LINKS - Explore popular categories

ENVATO TUTS+

About Envato Tuts+
Terms of Use
Advertise

JOIN OUR COMMUNITY

Teach at Envato Tuts+
Translate for Envato Tuts+
Forums

HELP

FAQ
Help Center

tuts+

28,004 **1,257** **39,815**
Tutorials   Courses   Translations

Envato.com   Our products   Careers   Sitemap

© 2019 Envato Pty Ltd. Trademarks and brands are the property of their respective owners.

Follow Envato Tuts+

Facebook Twitter   Pinterest