Selecting Contacts using device default contacts app in flutter.

# Contacts\_service package

Getting and working with contacts in your flutter app is as simple ringing up the contacts service package at <https://pub.dev/packages/contacts_service> to come work for you. With several static methods to get, add, update, delete contacts comfortably and a rich Contact class with nearly any contact property existing, it is no wonder that this package has a score rating of 98 on pub.dev (as at publishing this article) marking its health and popularity.

Only challenge with this package is that it does not manage app permissions itself. However, you easily overcome this by employing permission handler package <https://pub.dev/packages/permission_handler>. We’ll get into all that in a moment.

# New awesome functionality

After PR #100 was merged into the project (<https://github.com/lukasgit/flutter_contacts/pull/100>), we got a new functionality. If you want to select a contact, you no longer have to build an interface to display the contacts in the device and handle selection yourself. Yaay!

The `openDeviceContactPicker()` opens the native device contacts app and allows you search and select a contact, and returns your selection to your app. Asides doing that part of the work for you, users get to interact with their contacts using the same app they understand and are comfortable with, irrespective of device and platform.

Awesome isn’t it? Let’s build a simple demo together.

#picture

# Let’s make an example

Our objective – is to select a contact and get its details in our flutter app. We’ll display some of these details for our example.

First, let’s build the simple interface we’ll use.

## Building our company

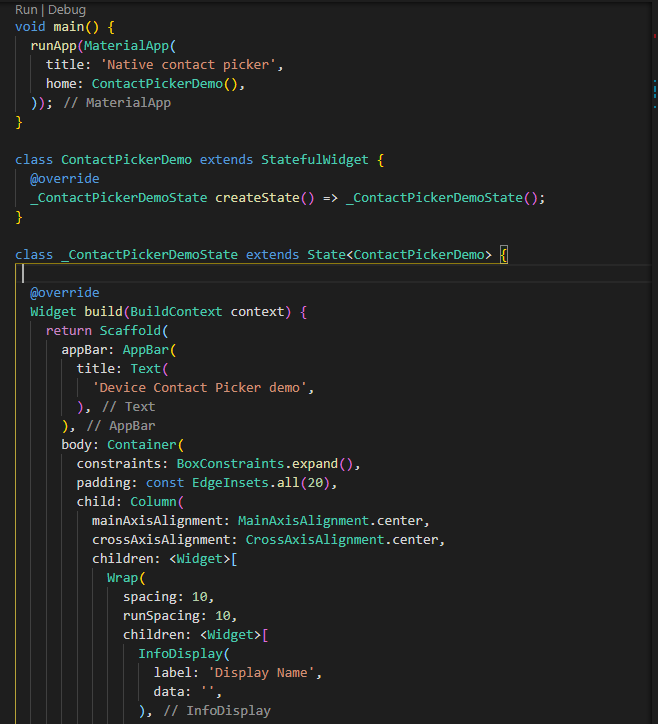
Create a new flutter project, either from your preferred IDE/Editor, or by running flutter create on the terminal.

`flutter create contact\_picker\_demo`

Let’s build a simple widget in our project to display details from our contacts to keep our code clean. The widget will take two strings: label and data. We will call it InfoDisplay.

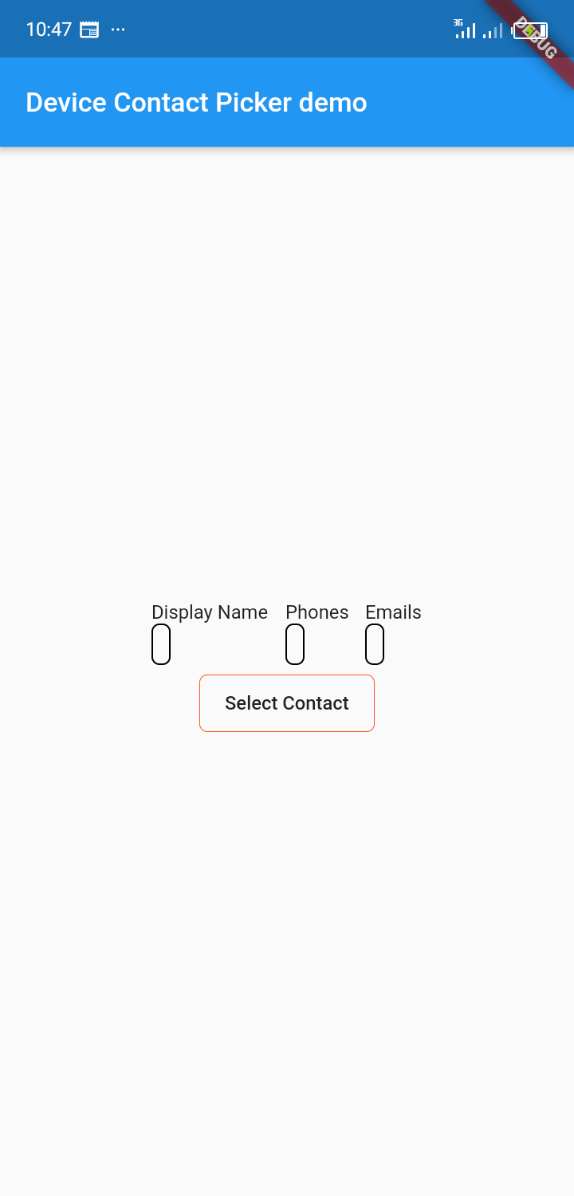


Now let us build the main layout. We use a stateful widget since the app will need to change state to display the selected contact every time it changes.





So far so good.



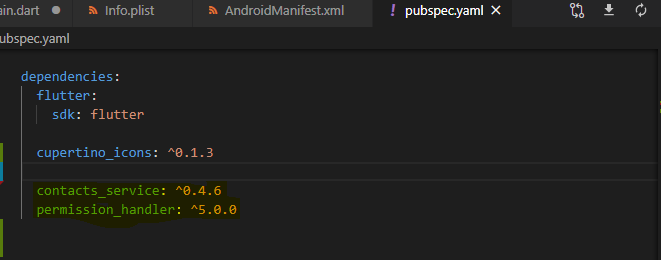
We’ve not assigned data to anything. We’ll do that once we return the selected contact to our app.

The Select Contact button does not do anything too. We’ll change that later.

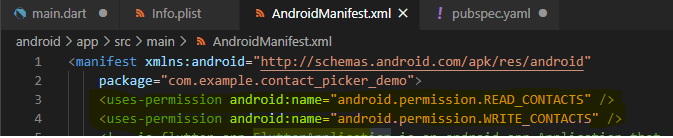
## Recruiting the packages

Let us get the packages into our app so we can begin working with them.

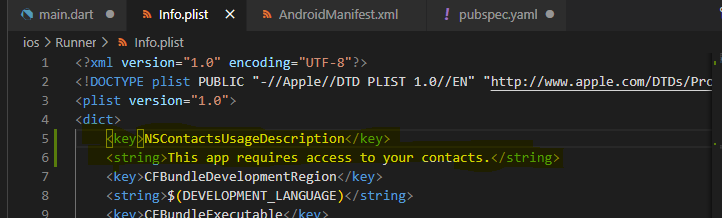
Depend on both packages in your pubspec.yaml file and run `flutter pub get`



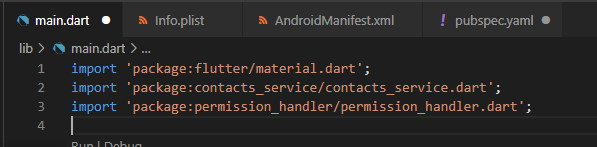
Following the readme of the package, we must add these lines to our AndroidManifest.xml file (./android/app/src/main/AndroidManifest.xml)



And make these additions to our info.plist file for iOS (./ios/Runner/Info.plist)



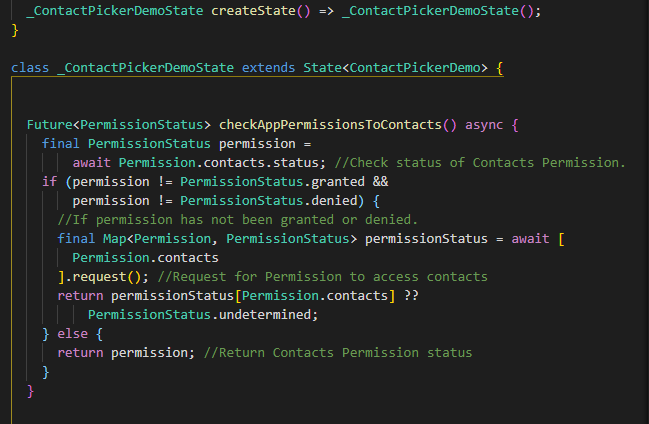
Nearly set! But we cannot use our packages without importing them. Let’s fix that in our main.dart file.



Next, let us handle permissions.

## Getting business license from the government: Handling permissions

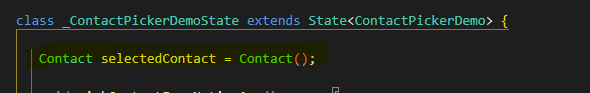
We’ll write a simple function to check the status of contact permission in our app. If permission is not granted, we’ll make a request for it. We’ll put this function in the state class, as well as every other thing going forward.



We can now go ahead to select a contact.

## Running our contacts business

Let’s define a new object of the Contact class to hold our selected contact.



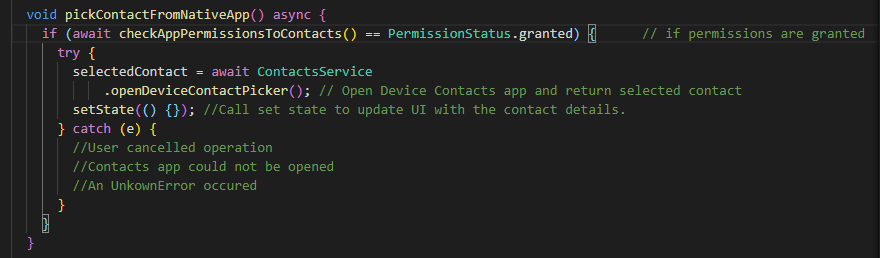
We will write a pretty straightforward function to get the selected contact here. We will use the function we wrote above to check for app permissions. If the permission to access contacts is granted, we will use the ContactsService to select a contact. We wrap the method call in a try and catch block to catch exceptions.

The openDeviceContactPicker throws three exception types.

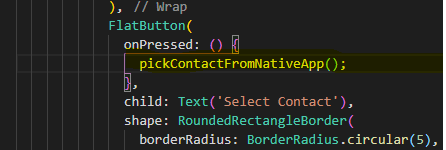
* FORM\_OPERATION\_CANCELLED: thrown when you navigate backwards without selecting a contact.
* FORM\_COULD\_NOT\_BE\_OPEN: thrown if for some reason, the contacts app on the device could not be opened.
* FORM\_OPERATION\_UNKNOWN\_ERROR: thrown if anything else went wrong.

You can handle these errors how you may, depending on what you need.

Note that if we call any of ContactsService methods without granted permissions, our app crashes. We don’t want that.

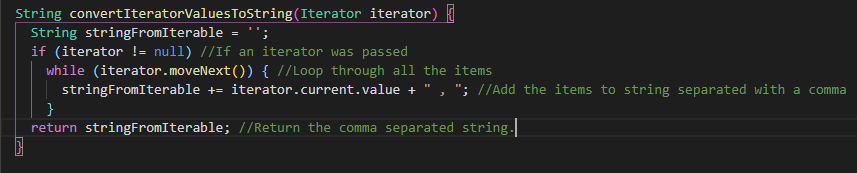


Now let’s call the function in our button. Find the button in the build method and add call the function.



If you run the app now and tap the button, surely you get your device contact form opened, and after you select one, we return to the app. Whatever your app needs contact for, you can now do that with the selectedContact object. We are just going to show the display name, phone numbers and emails. Shall we?

Phone numbers and emails are iterables and we need to show them as string. This function takes an iterator of iterables and makes a comma-separated string of all the elements.



Now let’s show some data from the contact on our UI. We find the InfoDisplay widgets we made earlier, and construct it with properties of `selectedContact` as data.



And voila! We’re done. Full code below.

#GIF

import 'package:flutter/material.dart';

import 'package:contacts\_service/contacts\_service.dart';

import 'package:permission\_handler/permission\_handler.dart';

void main() {

  runApp(MaterialApp(

    title: 'Native contact picker',

    home: ContactPickerDemo(),

  ));

}

class ContactPickerDemo extends StatefulWidget {

  @override

  \_ContactPickerDemoState createState() => \_ContactPickerDemoState();

}

class \_ContactPickerDemoState extends State<ContactPickerDemo> {

  Contact selectedContact = Contact();

  void pickContactFromNativeApp() async {

    if (await checkAppPermissionsToContacts() == PermissionStatus.granted) {      // if permissions are granted

      try {

        selectedContact = await ContactsService

            .openDeviceContactPicker(); // Open Device Contacts app and return selected contact

        setState(() {}); //Call set state to update UI with the contact details.

      } catch (e) {

        //User cancelled operation

        //Contacts app could not be opened

        //An UnkownError occured

      }

    }

  }

  Future<PermissionStatus> checkAppPermissionsToContacts() async {

    final PermissionStatus permission =

        await Permission.contacts.status; //Check status of Contacts Permission.

    if (permission != PermissionStatus.granted &&

        permission != PermissionStatus.denied) {

      //If permission has not been granted or denied.

      final Map<Permission, PermissionStatus> permissionStatus = await [

        Permission.contacts

      ].request(); //Request for Permission to access contacts

      return permissionStatus[Permission.contacts] ??

          PermissionStatus.undetermined;

    } else {

      return permission; //Return Contacts Permission status

    }

  }

  String convertIteratorValuesToString(Iterator iterator) {

    String stringFromIterable = '';

    if (iterator != null) //If an iterator was passed

      while (iterator.moveNext()) {

        //Loop through all the items

        stringFromIterable += iterator.current.value +

            " , "; //Add the items to string separated with a comma

      }

    return stringFromIterable; //Return the comma separated string.

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: Text(

          'Device Contact Picker demo',

        ),

      ),

      body: Container(

        constraints: BoxConstraints.expand(),

        padding: const EdgeInsets.all(20),

        child: Column(

          mainAxisAlignment: MainAxisAlignment.center,

          crossAxisAlignment: CrossAxisAlignment.center,

          children: <Widget>[

            Wrap(

              spacing: 10,

              runSpacing: 10,

              children: <Widget>[

                InfoDisplay(

                  label: 'Display Name',

                  data: selectedContact.displayName,

                ),

                InfoDisplay(

                    label: 'Phones',

                    data: convertIteratorValuesToString(

                        selectedContact.phones?.iterator)),

                InfoDisplay(

                    label: 'Emails',

                    data: convertIteratorValuesToString(

                        selectedContact.emails?.iterator)),

              ],

            ),

            FlatButton(

              onPressed: () {

                pickContactFromNativeApp();

              },

              child: Text('Select Contact'),

              shape: RoundedRectangleBorder(

                borderRadius: BorderRadius.circular(5),

                side: BorderSide(

                  width: 0.5,

                  color: Colors.deepOrange,

                ),

              ),

            )

          ],

        ),

      ),

    );

  }

}

class InfoDisplay extends StatelessWidget {

  InfoDisplay({@required this.label, @required this.data});

  final String label;

  final String data;

  @override

  Widget build(BuildContext context) {

    return Column(

      crossAxisAlignment: CrossAxisAlignment.start,

      mainAxisSize: MainAxisSize.min,

      children: <Widget>[

        Text(label ?? ''),

        Container(

          padding: const EdgeInsets.all(5),

          child: Text(data ?? '', maxLines: 5),

          // height: 30,

          decoration: BoxDecoration(

            borderRadius: BorderRadius.circular(5),

            border: Border.all(),

          ),

        ),

      ],

    );

  }

}

Take a look at the Contact class from the contacts\_service package to discover the other properties you have access to and explore the other methods of the contacts\_service package. Do enjoy coding in flutter.

#MajorE