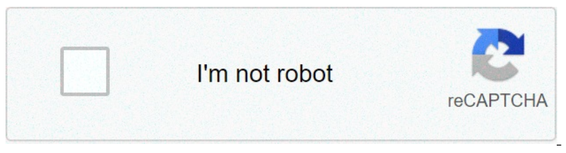

Continue

Flutter import java library

Imagine you have already completed a flutter project, and here the customer tells you that it has an Android project ready to add to the Flutter project. Here you have two ways. I completely rewrite the Android application to Flutter or import it into the Flutter project, which we will do. For example, I will use the application chat-bot. So open Android Studio and create a new fluttering project. Moreover, for more practicality, open the native Android part of the project in the Android module. In my case, I will add the chat bot chat application from the .aar file to the Modules tab we added a new module to our .jar / .aar Package Package project and on the dependency tab, we connected this form to our project. We have added a third-party Android project to our project. In Android / App / SRC / Main Edit AndroidManifest.xml in Android / Build.gradle Add JITPACK ALLPROJECTS {repository {Ã ¢ â,¬ | .. maven {url ' '}}} JITPACK controls the code, builds it and server build artifacts (vessel, AAR). In Android / App / Build.gradle add the path to our form and the addiction you need: dependencies {implaenzationÃ ¢ â,¬ | Ã ¢ â,¬ | Impensation project (route: ': stda-bot')} It is also necessary to configure the AndroidX support - ﬀ we fully integrated the chat- Bot in the Flutter project. Now we need to create a channel through which we will call our chat-bot from the Project Flutter. In Android / App / Main / Java / Com.Example / MainActivity create a channel and a method in which we call our Android application. Now change lib / main.dart: - Add import 'package: flutter / services.dart'; - Create the _GetMessage () method. - Call our method from FloatingActionButton. So, in our flutter project, we have successfully integrated and called a third-party Android project. Content content It is not practical to rewrite the entire application to flutter everything at once. For those situations, the flutter can be integrated into your existing fragmentary application, such as a library or form. That module can then be imported into the Android or iOS app (currently supported platforms) to make the UI part of your flutter app. O, just to execute the logic of the shared dart. In a few steps, you can bring the productivity and the expressiveness of the flutter into your app. Starting from Flutter V1.12, Add-to-App is supported for the basic scenario of the integration of a full screen fluctuation instance at a time per app. Currently it has the following limitations: multiple flutter libraries packaging in an application is not supported. The plugins used in add-to-apps on Android should migrate to the new Android plug-in bees, based on flutterplugin. The plugins that do not support flutterplugin could have unexpected behavior if they make hypotheses that are unsustainable in adding ad-to-app (how to assume that a flutter activity is always present). Starting from the V1.17, the Flutter module only supports AndroidX applications on Android. Starting from Flutter V1.26, adding experimenting experimentally by adding more instances of engines, screens or views on your flutter app. This can help integration scenarios like a hybrid navigation stack with native screens and mixed floating or a page with more partial flutter views. More than flutter instances allow each instance to maintain the independent application and the status of the user interface while using minimum memory resources. See More on the multiple flutter page. Supported functions Add to Auto-Build Android applications and import the fluttering module by adding a SDK flutter hook to the gradle script. Build your form to float in an archive Generic (AAR) for integration into its own build system and for better jepifier interoperability with AndroidX. API flutterengine to start and persist your flutter environment independently to attach a fluctuativeness / flutterFragment, etc. Etc. Android Studio Android / Flutter Co-editing and Creation / Import Wizard modules. The Java and Kotlin host apps are Flutter modules can use flutter plugins to interact with the platform. The Android plugins must be migrated to the V2 plugin APIs for the best additional correctness to the app. Starting from Flutter V1.12, most plugins managed by the flutter and flutterfire team have been migrated. Support for the US flutter debugging and hot reloading using the Flutter connection from IDES or the command line to connect to an app that contains flutter. Add to iOS Auto-Build Applications and import the Flutter module by adding a fluttering SDK hook to your cacapods and the XCode build phase. Build your fluttering module in a generic iOS framework for integration into your build system. API Flutterengine to start and persist your flutter environment independently to attach a FlutterViewController. Objectives-C and Supported Swift Host Apps. Flutter modules can use flutter plugins to interact with the platform. Support for the US flutter debugging and hot reloading using the Flutter connection from IDES or the command line to connect to an app that contains flutter. Look at our github sample repository add-to-app for sample projects in Android and iOS that import a fluttering module for UI. Start starting, consult our project integration guide for Android and iOS: use of the API after flutter is integrated into your project, consult our API usage guides at the following links: Flutter content can be incorporated into Your existing fragmenting Android application, as a Source Code Gradle SottoProject or as Aars. The integration flow can be done using the IDE Android Studio with the Flutter or manually plugin. Attention: your existing Android app can support architectures like MIPS or X86. Currently fluttering only supports the construction of bookcases compiled in advance (AOT) for X86_64, ARMEABI-V7A and ARM64-V8A. Consider using the Android ABI equipment Gradle API plugin to limit the architectures supported in your APK. Do this avoids a missing libflutter. Andreo Runtime, for example: Android {// ... deferoidconfig {ndk {// filter for architectures supported by flutter. 'ARMEABI-V7A' enabled, 'ARM64-V8A', 'X86_64'}}} The Flutter motor has an X86 and X86_64 version. When using an emulator in JUST-IN-TIME debug mode, the Flutter module still works correctly. Using Android Studio The Android Studio IDE is a convenient way to automatically integrate the Flutter module. With Android Studio, you can co-edit both your Android code and your flutter code in the same project. You can also continue to use the normal features of Intellij Flutter plugin such as the completion of the DART code, the hot top-up and the widget inspector. Additional flows to app with Android Studio are only supported on Android Studio 3.6 with the 42+ version of the Flutter plugin for Intellij. The integration of Android Studio also supports integration using a SottoCoject Gradle Code Source, rather than to use Aars. See below for more details on the distinction. Using the file> New> new module ¢ â,¬ | Menu in Android Studio in your existing Android project, you can create a new flutter module to integrate or select an existing Flutter module created previously. If you create a new module, you can use a wizard to select the form name, location and so on. The Android Studio plugin automatically configures your Android project to add the flutter module as an addiction and your app is ready to build. Note: To view changes automatically to the Android project from the IDE plugin, consider using the source control for your Android project before executing any A diff media of changes. Tip: By default, the project pane of your project is probably showing vision Ã ¢ â,¬ Å "AndroidÃ ¢ â,¬ â" ¢. If you can't see your new flutter filters in the project pane, make sure the project pane is set to display Ã ¢ â,¬ Å "Project FilesÃ ¢ â,¬ â" ¢, which shows all files without filtering. Your app now includes the flutter module as addiction. You can jump to the addition of a flutter screen to an Android app to follow next steps. Manual integration to integrate a flutter module with an existing Android application manually, without using the Android study plugin flutterÃ ¢ s, follow these steps: Create a Let s flutter module assume that you have an existing Android application in some / Path / MyApp and you want your flutter project as a brother: $ cd some / path / $ flutter create -t-t module com.Example My_flutter This creates a / my_flutter / flutter module project some / path with a little 'DART code to get started and a subfolder .android / hidden. The .android folder contains an Android project that can be helped to run the version of a standalone barebone of the module through flutter flutter race and ita s also a wrapper that helps bootstrap the flutter form integrated an Android library. Note: Add custom Android code to your existing project ApplicationA s or plugin, not the form in / .android. Changes in .android / your directory moduleÃ ¢ s Wona t appear in the existing Android project using the form. Do not check the source code of the / directory .android from ita s automatically generated. Before building the form of a new machine, run Get pubs flutter in the first my_flutter directory to regenerate the directory / .android before building the Android project using the Flutter form. Note: To avoid problems Dex fusion, flutter.androidPackage should not be identical to your Appa guest's name Java 8 engine requirement packet The Flutter Android uses Java 8 features. Before groped to link the project Flutter form to your host Android app, so that your Android host asserts the following compatibility source inside your Appa s build.gradle file, under the Android {} block, such example: Android {// ... compileOptions sourceCompatibility 1.8 targetCompatibility {1.8}} Add the Flutter module as a dependency Next, add the Flutter module as a dependency of your existing application in Gradle. There are two ways to achieve this goal. The AAR mechanism creates RAA Android generic as intermediaries that packets â â Flutter module. This is good when the app builders downstream donates t want to have installed the SDK Flutter. But, he adds another generation step if it builds frequently. The source code sub-mechanism is a convenient process of building a one-click, but it requires the Flutter SDK. This is the mechanism used by the plugin for Android Studio IDE. Option A - They depend on the Archives Android (AAR) This Flutter your library Packages option as a generic local Maven repository consists of the annual reports of activities and artifacts Poms. This option allows your team to build the host application without installing the SDK Flutter. It is therefore possible to distribute the artifacts from a local or remote repository. Let's assume you've built up a form at some Flutter / path / my_flutter, and then run: $ cd some / path / my_flutter $ flutter accumulation aar Then, follow the instructions on the screen to integrate. In particular, this command creates (by default all mode debug / profile / release) a local repository, with the following files: build / host / outputs / repo Ã Ã Ã Ã Ã com example Ã Ã Ã Ã my_flutter flutter_release aa 1.0 Ã Ã Ã Ã Ã Ã ¢ flutter_release-1.0.aar Ã Ã Ã Ã Ã Ã Ã Ã flutter_release-1.0.aar .md5 Ã Ã Ã Ã Ã Ã Ã Ã Ã Ã flutter_release -1.0.aar.sha1 Ã Ã Ã Ã Ã Ã Ã 1.0.pom flutter_release-It Ã Ã Ã flutter_release-1.0.pom .md5 Ã Ã Ã Ã Ã Ã flutter_release-1.0.pom.sha1 aa Maven-metadata.xml Ã Ã Ã Ã ¢ Maven- metadata.xml.md5 aa Maven-metadata.xml.sha1 Ã Ã Ã Ã Ã Ã Ã flutter_profile ... Ã Ã Ã Ã Ã Ã flutter_debug ... to depend on Annual activity, the host application must be able to find these files. To do this, edit app / build.gradle in your host application so that it includes the local repository and addiction: android {// ...} {repository maven {url 'some / path / my_flutter / build / guest / outputs / Repo '// This is related to the location of the Build.gradle file // if a relative path is used. } {URL Maven Maven }} Dependencies {// ... debugimplementation 'com.example.flutter_module: flutter_debug: 1.0' profilomplion 'com.example.flutter_module: flutter_profile: 1.0' religators 'com.example.frt_module: flutter_release: 1.0'} Important: if you â,¬ Â "¢ King Located in China, use a mirror site as https: // [a mirror site] /download.flutter.io rather than the domain storage.googleapis.com. See our use of flutter in China page for information on mirrors. Tip: You can also build an AAR for your flutter module in Android Studio using the Build menu> Flutter> Build Aar. Your app now includes the flutter module as addiction. You can follow the next steps in adding a flutter screen to an Android app. Option B - depends on the source code of the module This option allows a one-step build for the Android Project and Flutter project. This option is convenient when you work on both sides simultaneously and quickly iterano, but your team must install the SDK Flutter to create the host app. Tip: By default, the Host app provides the Gradle app project. To change the name of this project, set Flutter.HostappProjectName to the Blitter Module file. File.properties. Finally, include this project in the host app settings file. Gradel mentioned below. Include Flutter module as sottoproject dell'Host.Gradle Settings app: // Include the App Host project. Include ': app' // has taken the existing content of existing content (new bond ([Gradle: this]))) // new currency (new file (// new settingsdir.parentfile, // new 'my_flutter / .android / included_flutter.groovy '// new)) // new assuming that my_flutter is a brother in MyApp. The binding and script assessment allows the Fluctuation module to include itself (such as: Flutter) and any Flutter plugin used by the module (such as: Packages_Info ,: Video_Player, etc.) In the settings assessment context. Gadladle. Enter a implementation dependence from the Flutter module from your app: Dependencies {Implementation Project (': Flutter')} The App now includes the Flutter module as dependence. You can follow the next steps in adding a flutter screen to an Android app. app.