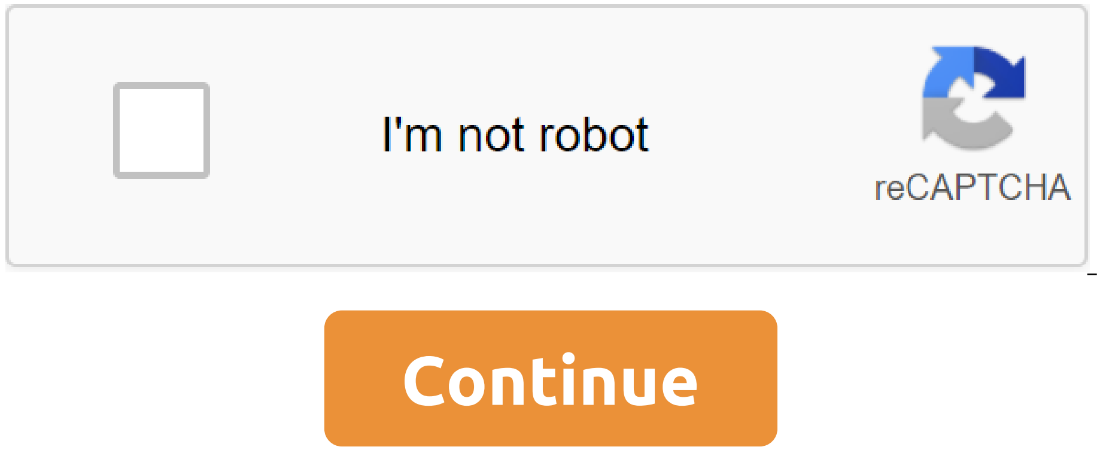
Android developer speech recognition example



SpeechRecognizer expands java.lang.Object ↳ android.speech.Speech.SpeechRecognizer This class provides access to the speech recognition service. This service allows you to access speech recognition. Do not instant this class directly, instead, call SpeechRecognizer'createSpeechRecognizer (Context). This class's methods should only be called from the main flow of the application. Implementing this API is likely to stream audio to remote servers to perform speech recognition. As such, this API is not designed for continuous recognition, which will consume a significant amount of battery and bandwidth. Please note that the app must have Manifest.permission.RECORD_AUDIO permission to use this class. From the java.lang.Object Object class, the clone creates and returns a copy of this object. boolean (Object obj) indicates whether any other object is equal to this. invalid completion () Is called by the garbage collector at the facility when the garbage collection determines that there are no more references to the object. The final class of the getClass returns the time class of the subject. int hashCode () Returns the hash code value to the object. the final invalid to notify () will wake up one thread that is waiting on the monitor of this object. ToString returns the view of the object line. The final expectation of emptiness (long time out, int nanos) triggers anticipation of the current thread until another thread triggers the notification method () or the notifyAll method for that object, or some other thread interrupts the current thread, or a certain amount of real time has passed. The final expectation of emptiness (long time) triggers the wait for the current thread until another notification method () or notifyAll method has been triggered for that object, or a certain amount of time has passed. the final expectation of emptiness () causes the current thread to wait until another thread triggers the notification method () or the notifyAll method for that object. Permanent public static final lines CONFIDENCE_SCORES key used to produce a float array from a set of RecognitionListener-onResults (Bundle) and RecognitionListener-onPartialResults (Bundle). The array should be the same size as The ArrayList presented in the RESULTS_RECOGNITION, and must contain values of 0.0 to 1.0, or -1, to represent an inaccessible confidence estimate. Trust values close to 1.0 indicate high confidence (the speech detector is confident that the recognition result is correct), while values close to 0.0 indicate low confidence. This value is optional and cannot be provided. Permanent value: confidence_scores static final int ERROR_AUDIO audio error. Permanent value: 3 (0x0000003) public static final int ERROR_CLIENT other customer hand errors. Permanent value: 5 5 Public static final int ERROR_INSUFFICIENT_PERMISSIONS Insufficient Resolution Permanent Value: 9 (0x0000000) public static finale int ERROR_NETWORK other errors related to the network. Permanent value: 2 (0x0000002) public static ERROR_NETWORK_TIMEOUT the time-out network operation. Permanent value: 1 (0x0000001) public static finale int ERROR_NO_MATCH the result of recognition does not correspond. Permanent value: 7 (0x0000007) public static final int ERROR_RECOGNIZER_BUSY RecognitionService is busy. Permanent value: 8 (0x0000008) public static final int ERROR_SERVER Server sends error status. Permanent value: 4 (0x00000004) public static final int ERROR_SPEECH_TIMEOUT No Speech entering Permanent value: 6 (0x0000006) Public methods of public emptiness to destroy () Destroys the SpeechRecognizer object. public static boolean isRecognitionAvailable (ContextUal Context) checks whether speech recognition is available in the system. If this method returns false, SpeechRecognizer'createSpeechRecognizer (Context) will fail. Context Context Options: By which SpeechRecognizer will be created Returns boolean true if recognition is available, false otherwise public voids setRecognitionListener (RecognitionListener listener) Sets the listener who will receive all callbacks. Previous unfinished commands will be performed with an old listener, while any next command will be performed with a new listener. RecognitionListener Listener Options: The listener who receives all the callbacks from the created SpeechRecognizer, it should not be zero. public emptiness startListening (RecognizerIntent) Begins to listen to the speech. Please note that setRecognitionListener (android.speech.RecognitionListener) must be called in advance, otherwise no notifications will be received. Recognizable Intent Options: Contains recognition options that need to be performed. Intention may also contain additional services, see if these values are not explicitly set, the defaults will be used by the recognizer. public emptiness stopListening () Stops listening to speech. The speech captured so far will be recognized as if the user has stopped speaking at that moment. Note that in the default case, this does not need to be called, as the final point of speech will automatically stop the listening to the recognized when it determines that the speech is complete. However, you can manipulate the final points settings directly with the additional intentions defined in RecognizerIntent, in which case sometimes you can call manually to stop listening Please note that setRecognitionListener (android.speech.RecognitionListener) must be called in advance, otherwise no notifications will be received. Voice actions are an important part of the wearable experience. They allow users to act loud and fast. Google's Wear OS provides two types of voice action: actions: These voice actions are based on tasks and are built into the Wear platform. You filter them in the actions you want to start when the voice action is pronounced. Examples include Take note or set an alarm. The app's data are based on the app, and you announce them in the same way as the launcher icon. Users say: Start your app name to use these voice actions and the actions you specify begins. The Declare The Voice Action System's Wear OS platform provides several voice intentions based on user actions, such as Take a note or set an alarm. This allows users to say what they want to do and let the system figure out the best action to start. When users talk about voice action, your app can filter for the intent that starts to start the action. If you want to start a service to do something in the background, show up as a visual signal and start serving in activities. Make sure to call finish () when you want to get rid of the visual signal. For example, for the team, take note announce this filter of intent to start an activity called MyNoteActivity: qlt;activity android:nameMyNoteActivitygt; qlt;action-action android:name.android.intent.action.send'lt;lt;/action-category-android:name'com.google.android.voicesearch.SELF_NOTE Here's a list of voice intentions supported by the Wear platform: Name the example of phrases Of Intention Call car/taxi OK Google, give me a taxiOK Google, call me a car Action com.google.android.gms.actions.RESERVE_TAXI_RESERVATION Take note OF GOOD Google, take note of Google's OK, Note to Yourself Action Android.action .SEND Category com.google.android.voicesearch.SELF_NOTE Extra android.content.Intent.EXTRA_TEXT - a line with a body note Set the alarm set OK Google, set the alarm for 8am Google, wake me up at 6 tomorrow Action android.intent.action.SET_ALARM Extras android.provider.AlarmClock.EXTRA_HOUR - integer with an hour of alarm. android.provider.AlarmClock.EXTRA_MINUTES - integer with minute alarm (these 2 extras are optional, either not or both are provided) Set a timer Ok Google, Set a timer for 10 minutes Action android.intent.action.SET_TIMER Extra android.provider.AlarmClock.EXTRA_LENGTH - a whole in the range of 1 to 86,400 (number of seconds in 24 hours), representing the duration of the timer Running stopwatch Ok Google, start stopwatch Action com.google.android.wearable.action.STOPWATCH Start/Stop bike ride OK Google, Bike ride OK Google, start your bike rideOK Google, stop cycling Action vnd.google.fitness.TRACK Mime Type vnd.google.fitness.activity/biking Extras actionStatus - a line with the value of ActiveActionStatus at launch. Run /Stop run Good Google, track my startup OK Google, start working OK Google, stop working Action vnd.google.fitness.TRACK MimeType MimeType Extras actionStatus - a line with the value of ActiveActionStatus at launch and completedActionStatus when stopping Start/Stop training OK Google, start training OK Google, track my workouts OK Google, stop training Action vnd.google.fitness.google.fitness.activity/other Extras actionStatus OK Google, what's my bpm? Action vnd.google.fitness.VIEW Mime Type vnd.google.fitness.data_type/com.google.heart_rate.bpm Show Pitch Count Good Google, How Many Steps Have I Made? OK Google, what's my pitch count? Action vnd.google.fitness.VIEW Type vnd.google.fitness.data_type/com.google.step_count.cumulative For registration documentation for platform intentions and access to additional information contained in them, see Announce the voice actions provided by the app if none of the platform's voice intentions work for you, you can start your applications directly with The Voice Action Start MyActivityName. Registration for the Start action is the same as the registration of the launcher icon on the portable. Instead of requesting an app icon in the launcher, the app asks for voice action. To specify the text to say after Start, specify the label attribute for the action you want to start. For example, this intention filter recognizes the voice action of Start MyRunningApp and launches StartRunActivity. Getting a free speech input form In addition to using voice action to run up activities, you can also call the built-in speech recognition system activities to receive speech input from users. Android:name'StartRunActivity android: ?label MyRunningApp Action:android.intent.action.MAIN/action/gt; qlt;category android:name.name.category.launcher In the app, you call startActivityForResult using ACTION_RECOGNIZE_SPEECH action. This triggers speech recognition activities and you can process the result in onActivityResult. Private const val SPEECH_REQUEST_CODE No 0 ... / Create an intention that can start speech recognition activities private amusing displaySpeechRecognizer () - val intent (RecognizerIntent.ACTION_RECOGNIZE_SPEECH RecognizerIntent.EXTRA_LANGUAGE_MODEL). RecognizerIntent.LANGUAGE_MODEL_FREE_FORM) / Start the action, the intention will be populated by a text runActivityForResult (intention, SPEECH_REQUEST_CODE) This callback is called when the speech recognition is returned. Here you process the intention and extract the text of the speech from the intention. redefine onActivityResult (requestCode: Int, Int, Int, data: Intention) - SPEECH_REQUEST_CODE if (requestCode - SPEECH_REQUEST_CODE - resultCode - Activity.RESULT_OK) - val spokenText: String? - data.getStringArrayListExtra (RecognizerIntent.EXTRA_RESULTS &gt;). Create an intention that can start speech recognition by private invalid displaySpeechRecognizer () - Intention - a new intention (RecognizerIntent.ACTION_RECOGNIZE_SPEECH); intent.putExtra (RecognizerIntent.EXTRA_LANGUAGE_MODEL, RecognizerIntent.LANGUAGE_MODEL_FREE_FORM); The beginning of the action, the intention will be settled with the launch of the text of the speechForActivityForResult (intention, SPEECH_REQUEST_CODE); Here you process the intention and extract the text of the speech from the intention. @Override protected void onActivityResult (int requestCode, int resultCode, Intent Data) - if (requestCode - SPEECH_REQUEST_CODE - resultCode - RESULT_OK) - results of the list - data.getStringArrayListExtra (RecognizerIntent.EXTRA_RESULTS); String conversationtext and results.get (0); Do something with spokenText and super.onActivityResult (requestCode, resultCode, data); Also refer to the following related resources: OS Design Principles